

AD-A061 096

COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM. (QUICK). PRO--ETC(U)
1978

F/G 5/2

UNCLASSIFIED

CCTC-CSM-MM-9-77-V1-CHG-1

NL

1 OF 3
ADA
061096





DEFENSE COMMUNICATIONS AGENCY

COMMAND AND CONTROL
TECHNICAL CENTER

WASHINGTON, D. C. 20301

12

SC

LEVEL III

IN REPLY
REFER TO: C314

AD A061096

TO: RECIPIENTS

SUBJECT: Change 1 to Program Maintenance Manual
Volume I, Data Management Subsystem

CSM-MM-9-77-VOL-1-chg-1

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations. Pages 383 and 384 are included to correct a printing error.
2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.
3. When this change has been posted, make an entry in the Record of Changes.

FOR THE DIRECTOR

Enclosures
Change 1 pages

J. DOUGLAS POTTER
Assistant to the Director
for Administration

6 The CCTC Quick-Reacting General War
Gaming System. (Quick). Program
Maintenance Manual. Volume I.
Data Management Subsystem.

11 1978

Change I

12 267 p.

DDC
RECEIVED
NOV 9 1978

W D

ADDITIONAL TO	
OTIS	State Section <input checked="" type="checkbox"/>
OSG	State Section <input type="checkbox"/>
UNCLASSIFIED	<input type="checkbox"/>
IDENTIFICATION	
Ref: Basic Doc. - AD-A054377	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist. AVAIL. OR/ SPECIAL	
A	

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

DDC FILE COPY

AD A054377

409658

EFFECTIVE PAGES - JULY 1978

This list is used to verify the accuracy of CSM MM 9-77, Volume I after change 1 pages have been inserted. Original pages are indicated by the letter 0, and change 1 pages by the numeral 1.

<u>Page No.</u>	<u>Change No.</u>	<u>Page No.</u>	<u>Change No.</u>
Title Page, Part I	0	169-172	1
ii	0	173	0
iii-viii	1	174	1
ix-x	0	175	0
xi	1	176	1
1-3	0	177-434	0
4	1	434.1-434.2	1
5-14	0	434.3-434.4	0
15-16	1	Title Page, Part II	0
17	0	ii	0
18	1	iii-ix	1
19-29	0	x	0
30	1	xi	1
31-35	0	435-438	0
36-41	1	439	1
41.1-41.2	1	440-444	0
42-45	0	445	1
46-56	1	446-546	0
57-75	0	547-548	1
76-79	1	549-572	0
79.1-79.4	1	572.1-572.6	1
80-88	0	573-580	0
89-90	1	581	1
90.1-90.2	1	582-588	0
91	1	589	1
92	0	590	0
93	1	591-593	1
94-99	0	593.1-593.2	1
100-101	1	594-595	1
102	0	596-675	0
103-130	1	676	1
131-147	0	676.1-676.38	1
148-150	1	677-718	0
151-152	0	719	1
153	1	720	0
154	0	721	1
155-156	1	722-742	0
157-158	0	743-744	1
159-166	1	745-855	0
167-168	0	856-857	1

<u>Page No.</u>	<u>Change No.</u>
857.1-857.6	1
858-895	0
896-901	1
901.1-901.6	1
902-910	1
910.1-910.2	1
911	1
911.1-911.2	1
912-919	1
919.1-919.6	1
920-921	0
921.1-921.6	1
922	0
923-924	1
925-926	0

78 10 31 069

CONTENTS

Part I

Section	Page
ACKNOWLEDGMENT.....	ii
ABSTRACT.....	xi
1 GENERAL.....	1
1.1 Purpose.....	1
1.2 Program Environment - QUICK System Overview.....	1
1.3 Data Management Subsystem Overview.....	5
1.3.1 QUICK Integrated Data Base.....	5
1.3.2 QUICK Generalized Text English Commands.....	5
1.3.3 QUICK System Central Operations Processor (COP).....	6
1.3.4 Data Management Modules.....	6
1.3.5 General Utilities.....	6
1.4 Computer Software Environment and Programming Language.....	6
1.5 Equipment Environment.....	7
1.6 Organization of Maintenance Manual, Volume I.....	7
2 INTEGRATED DATA BASE.....	9
2.1 Purpose.....	9
2.2 Concept of Operation.....	9
2.2.1 IDS Records.....	9
2.2.2 Reference Code.....	9
2.2.3 IDS Chains.....	11
2.2.4 IDS Record Classes.....	11
2.2.4.1 Primary Records.....	11
2.2.4.2 Secondary Records.....	13
2.2.4.3 CALC Record.....	13
2.2.5 Record and Chain Tables.....	13
2.2.6 Chain Order.....	14
2.2.7 Match Keys.....	14
2.3 Functional Description.....	14
2.3.1 Entry Points for IDS Processing.....	14
2.3.1.1 Entry OPNIDS.....	17
2.3.1.2 Entry CLZIDS.....	17
2.3.1.3 Entry DIRECT.....	17
2.3.1.4 Entry NEXTTT.....	17
2.3.1.5 Entry HEAD.....	17
2.3.1.6 Entry STORE.....	17
2.3.1.7 Entry RETRV.....	17
2.3.1.8 Entry MODFY.....	17
2.3.1.9 Entry DLETE.....	18
2.3.1.10 Entry CLEANP.....	18
2.3.1.11 Error Recovery.....	18

Section	Page
2.3.2 Entry Points for Header Creation and Identification.....	18
2.3.2.1 Entry HDPUT.....	18
2.3.2.2 Entry HDFND.....	18
2.3.3 Input Instruction Code Processing.....	19
2.3.3.1 Verb and Adverb Instruction.....	19
2.3.3.2 Input Clause Types.....	19
2.3.3.3 Input Instruction Codes.....	21
2.3.3.4 Input Instruction Format.....	22
2.3.3.4.1 Miscellaneous Instructions.....	22
2.3.3.4.2 Logical Instructions.....	22
2.3.3.4.3 Instructions Using Internal Variable.....	29
2.3.3.4.4 The General Instruction Format.....	29
2.3.3.5 Input Instructions Code Entry Points....	29
2.3.3.5.1 INSGET.....	29
2.3.3.5.2 INSPUT.....	29
2.3.3.5.3 INSFLS.....	29
2.3.3.5.4 INSDEL.....	29
2.3.3.6 Instruction Code Example.....	29
2.3.4 Entry Points for Input Line Processing.....	30
2.4 QUICK's Data Base Structure.....	30
2.4.1 Scenario Data Structure.....	36
2.4.1.1 Target Data Structure.....	36
2.4.1.2 Weapon Data Structure.....	36
2.4.1.3 Geographic Data Structure.....	40
2.4.1.4 Sortie Data Structure.....	40
2.4.2 Organization Data Structure.....	40
2.4.2.1 Data Organization Index.....	40
2.4.2.2 Assignment Table.....	40
2.4.2.3 Miscellaneous Organizational Data.....	40
2.4.3 Data Base Record Content.....	44
2.4.3.1 Primary Records.....	44
2.4.3.2 Secondary Records.....	44
3 CENTRAL OPERATIONS PROCESSOR.....	57
3.1 Purpose.....	57
3.2 Input.....	57
3.3 Output.....	57
3.4 Concept of Operation.....	57
3.5 Identification of Subroutine Functions.....	58
3.5.1 Data Base Interface.....	58
3.5.2 Text English Syntax Analysis.....	58
3.5.3 Input Translation.....	60
3.5.4 Module Execution.....	60
3.5.5 Organizational Data Initialization.....	60
3.6 Common Blocks.....	60
3.7 Main Routine of COP.....	62
3.7.1 Subroutine BANNER.....	65
3.7.2 Subroutine ERPROC.....	68

Section	Page
3.7.3 Subroutine HDFND.....	70
(Entry HDFND)	
(Entry HDPUT)	
3.7.4 Subroutine INICOP.....	76
3.7.4.1 Subroutine INPRIN.....	79.1
(Entry INPRIN)	
(Entry LGPRIN)	
(Entry ERPRIN)	
3.7.5 Subroutine INSPUT.....	80
(Entry INSPUT)	
(Entry INSNT)	
(Entry INSFLS)	
(Entry INSDEL)	
(Entry INSGET)	
3.7.6 Subroutine MODGET.....	89
3.7.7 Subroutine QDATA.....	91
(Entry OPNIDS)	
(Entry CLEANP, CLZIDS and DIRECT)	
(Entry STORE)	
(Entry RETRV)	
(Entry NEXTTT)	
(Entries HEAD, MODFY, and DELETE)	
3.8 Subroutine BOOT.....	101
3.8.1 Subroutine DCTFND.....	131
3.8.2 Subroutine MNMFND.....	134
3.8.3 Subroutine NUMFND.....	136
3.8.4 Subroutine RNMFND.....	139
3.8.5 Subroutine SEEKER.....	141
3.8.6 Subroutine STRMAK.....	143
3.9 Subroutine ERRFND.....	145
3.9.1 Subroutine LNGSTR.....	148
3.9.2 Subroutine SYNTAX.....	153
3.9.3 Subroutine TABINS.....	178
3.9.4 Subroutine WEBSTR.....	185
3.10 Subroutine INPTRN.....	188
3.10.1 Subroutine DELTAB.....	227
3.10.2 Subroutine INMATH.....	229
(Entry INMATH)	
(Entry INUMB)	
(Entry INATT)	
(Entry INALPH)	
3.10.3 Subroutine LINEIO.....	236
(Entry LINEIO)	
(Entry LINGET)	
(Entry LINPUT)	
3.10.4 Subroutine PARLEV.....	241
3.10.5 Subroutine TABGET.....	243

Section		Page
4	DATA MODULE.....	247
4.1	Purpose.....	247
4.2	Input.....	247
4.3	Output.....	247
4.4	Concept of Operation.....	247
4.4.1	Retrieval Schemes.....	247
4.4.1.1	The Get Header Instruction.....	248
4.4.1.2	The Chain Next Instruction.....	248
4.4.1.3	The Chain Master Instruction.....	248
4.4.1.4	The Return Instruction.....	248
4.4.1.5	Retrieval Supporting Subroutines.....	249
4.4.2	Primary Header.....	249
4.4.3	Determining a Record Type Set from a List of Attributes.....	249
4.4.4	Data Queues.....	250
4.5	Identification of Subroutine Functions.....	251
4.5.1	CREAAT.....	251
4.5.2	CHANGE.....	251
4.5.3	DELETE.....	251
4.6	Common Blocks.....	251
4.7	Subroutine ENTMOD.....	258
4.7.1	Subroutine VALPUT.....	260
	(Entry VALPUT)	
	(Entry VALGET)	
	(Entry VALDEL)	
4.8	Subroutine CHANGE.....	265
4.8.1	Subroutine DESSCH.....	309
4.8.2	Subroutine NXTDES.....	316
4.9	Subroutine CREAAT.....	318
4.10	Subroutine DELETE.....	369
5	EDITDB MODULE.....	383
5.1	General Purpose.....	383
5.2	Input.....	383
5.3	Output.....	383
5.4	Concept of Operation.....	383
5.5	Identification of Subroutine Functions.....	383
5.5.1	Subroutine COUNTS.....	383
5.5.2	Subroutine GENEDIT.....	384
5.5.3	Subroutine BUILDTAB.....	384
5.5.4	Subroutine NORMAL.....	384
5.5.5	Subroutine PROCEDIT.....	384
5.6	Edit Internal Common Blocks.....	384
5.7	Subroutine ENTMOD.....	387
5.8	Subroutine COUNTS.....	390

Section	Page
5.9 Subroutine GENEDIT.....	399
5.9.1 Subroutine BUILDTAB.....	401
5.9.2 Subroutine FORMLOC.....	405
5.9.3 Subroutine SETFLD.....	407
5.9.4 Subroutine SWITH.....	412
(Entries SWITH and SWHERE)	
5.10 Subroutine NORMAL.....	417
5.11 Subroutine PROCEDIT.....	419
5.11.1 Subroutine XWITH	424

Part II

6	REPORT MODULE.....	435
7	SAVE AND RESTORE MODULE (SRM).....	573
8	EXTERNAL INTERFACE MODULE (EIM).....	581
9	GENERAL UTILITIES.....	677
Appendix		
A.	COP EXTERNAL COMMON BLOCKS.....	891
B.	EXECUTABLE JOB CONTROL LANGUAGE (JCL) QUICK SYSTEM..	895
C.	PERFORM PROGRAM.....	909
DISTRIBUTION.....		434.1
DD Form 1473.....		434.3

ILLUSTRATIONS (PART I)

Figure		Page
1	Major Subsystems of the QUICK System.....	3
2	Procedure and Information Flow in QUICK/HIS 6000....	4
3	Integrated Data Line Format.....	10
4	Example of Hierarchically Structured Data.....	12
5	Record Linkage Through Chains.....	12
6	Scenario Data Structure.....	37
7	Target Data Structure.....	38
8	Weapon Data Structure	39
9	Geographic Data Structure.....	41
9.1	Sortie Data Structure.....	41.1
10	Data Organization Index.....	42
11	Assignment Table.....	43
12	Miscellaneous Organizational Data.....	45
13	Program COP.....	63
14	Subroutine Banner.....	66
15	Subroutine ERPROC.....	69
16	Subroutine HDFND.....	71
17	Subroutine INICOP.....	77
17.1	Subroutine INPRIN.....	79.3
18	Subroutine INSPUT.....	82
19	Subroutine MODGET.....	90
20	Subroutine QDATA.....	92
21	Subroutine BOOT.....	105
22	Subroutine DCTFND.....	132
23	Subroutine MNMFND.....	135
24	Subroutine NUMFND.....	137
25	Subroutine RNMFND.....	140
26	Subroutine SEEKER.....	142
27	Subroutine STRMAK.....	144
28	Subroutine ERRFND.....	146
29	Subroutine LNGSTR.....	149
30	Subroutine SYNTAX.....	155
31	Subroutine TABINS.....	179
32	Subroutine WEBSTR.....	186
33	Subroutine INPTRN.....	190
34	Subroutine DELTAB.....	228
35	Subroutine INMATH.....	231
36	Subroutine LINEIO.....	237
37	Subroutine PARLEV.....	242
38	Subroutine TABGET.....	244
39	Subroutine CREAAT.....	252
40	Subroutine CHANGE.....	255
41	Subroutine ENTMOD (DATA).....	259
42	Subroutine VALPUT.....	261
43	Subroutine CHANGE: Step One.....	266
44	Subroutine CHANGE: Step Two.....	276
45	Subroutine CHANGE: Step Three.....	286
46	Subroutine CHANGE: Step Four.....	288

ABSTRACT

↓
The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, provide output summaries, and produce tapes to simulator subsystems external to QUICK. QUICK has been programmed in FORTRAN for use on the CCTC HIS 6000 computer system.

The QUICK Maintenance Manual consists of four volumes: Volume I, Data Management Subsystem; Volume II, Weapon/Target Identification Subsystem; Volume III, Weapon Allocation Subsystem, Volume IV, Sortie Generation Subsystem. The Maintenance Manual complements the other QUICK Computer System Manuals to facilitate application of the war gaming system.

This volume, Volume I, in two parts, provides the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the modules (programs) and sub-routines of the Data Management subsystem. Companion documents are:

↑
a. USERS MANUAL

Computer System Manual CSM UM 9-77, Volume I

Computer System Manual CSM UM 9-77, Volume II

Computer System Manual CSM UM 9-77, Volume III

Computer System Manual CSM UM 9-77, Volume IV

Provides detailed instructions for applications of the system

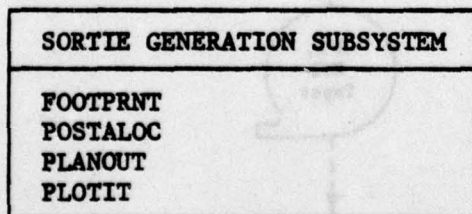
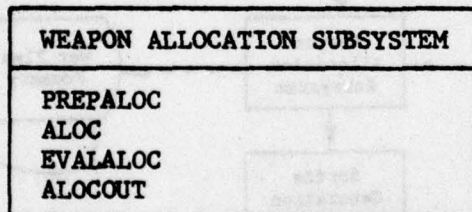
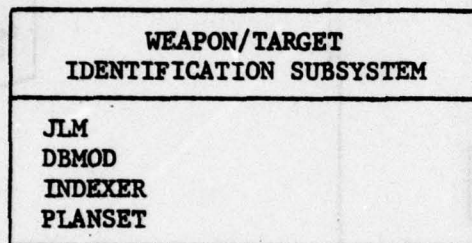
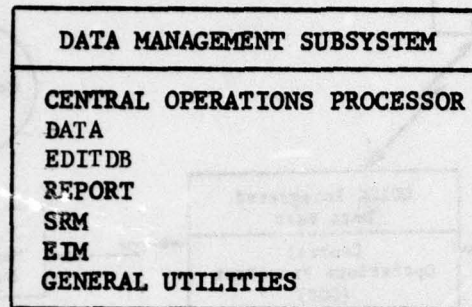
b. TECHNICAL MEMORANDUM

Technical Memorandum TM 153-77

Provides a nontechnical description of the system for senior management personnel

SUBSYSTEMS

FUNCTIONAL PARTS



EXECUTIVE SOFTWARE

DATA BASE PREPARATION

PLAN GENERATION

Figure 1. Major Subsystems of the QUICK System

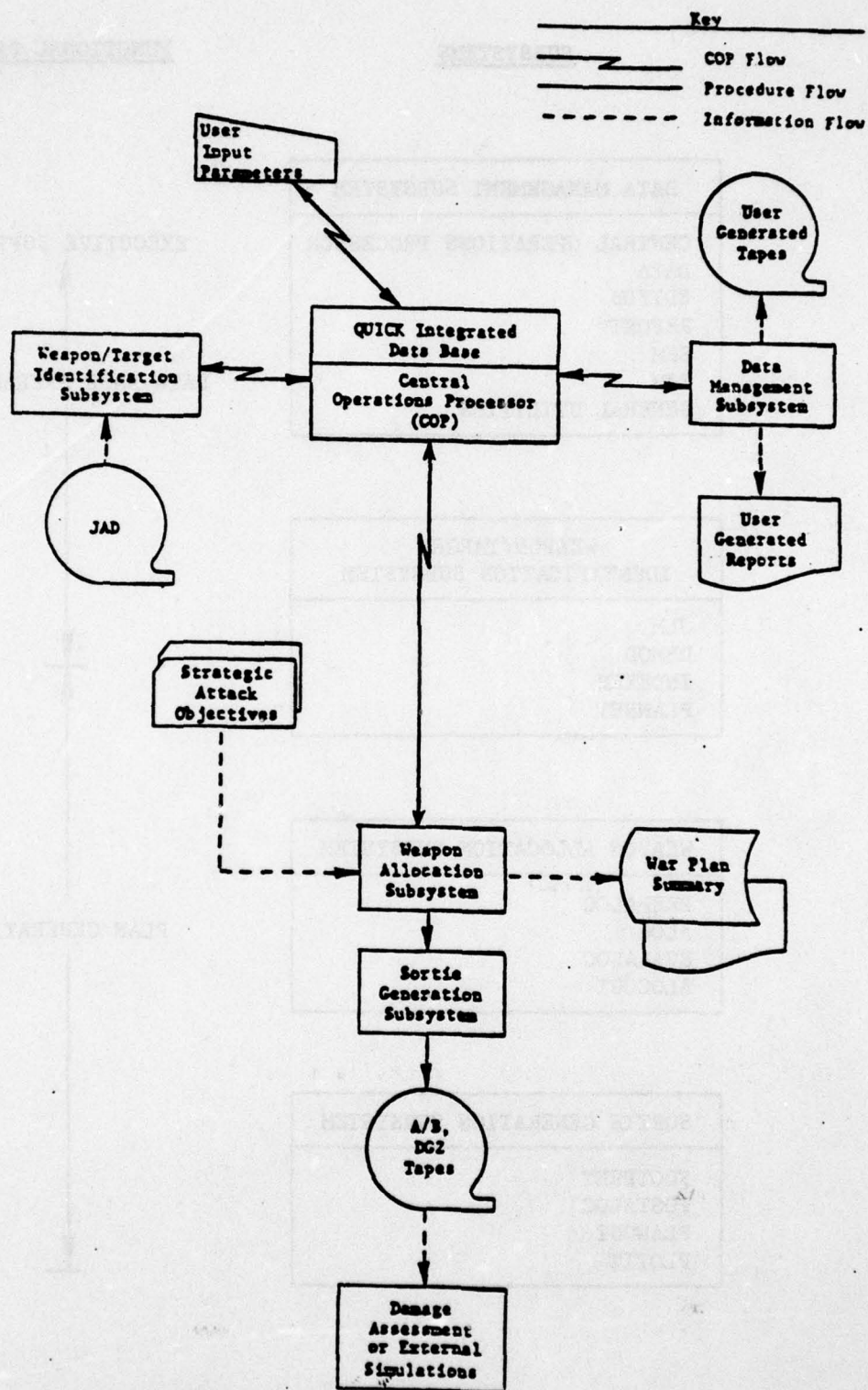


Figure 2. Procedure and Information Flow in QUICK/HIS 6000

Table 1. COP Entry Points (Part 1 of 2)

<u>ENTRY POINT NAME</u>	<u>ARGUMENTS</u>	<u>DESCRIPTION</u>
CLEANP	(none)	IDS Clean Point
CLZIDS	(none)	Close IDS File
DIRECT	(none)	Retrieves array IDS record based on its binary reference code stored in common C10
DLETE	Record Type Name	Deletes current record of type named
ERPRIN	(none)	Process input line for error message
HDFND	BCD reference Code, CLASS value, SIDE value, Record Type Name	Finds BCD reference code, given values for CLASS, SIDE and/or Record Type Name
HDPUT	BCD reference Code, CLASS value, SIDE value, Record Type Name	Adds New Header of Type named with given values for CLASS and SIDE
HEAD	Chain Name	Retrieves master of chain named
INPRIN	(none)	Process input line normally
INSDEL	(none)	Deletes all input tables
INSFLS	(none)	Assures that any additions to input tables are recorded
INSGET	Array to contain output, Index of first item to retrieve, Number of items to retrieve	Obtains input instructions from input tables
INSPUT	Array which contains items to be inserted, Index to input table (should be set to start point minus one, will be returned as end point), Number of items to add	Inserts items in input tables
LGPRIN	(none)	Process input line from long string

Table 1. (Part 2 of 2)

<u>ENTRY POINT NAME</u>	<u>ARGUMENTS</u>	<u>DESCRIPTION</u>
MODIFY	Record Type Name	Modifies current record of type named to reflect current values in common
NEXTTT	Chain Name	Retrieves next record of chain named
OPNIDS	(none)	Opens IDS file
RETRV	Record Type Name	Retrieve record of type named (should be used only for primary or CALC records)
STORE	Record Type Name	Store new record of type named

entry points is C30. Following each call that affects a particular record its binary reference code is stored in word 1 of C10 and its record type number is stored in word 4 of C10.

2.3.1.1 Entry OPNIDS. This is a one time call to open the IDS data file for processing. This entry point may only be called once in an activity.

2.3.1.2 Entry CLZIDS. This entry point closes the IDS data file. Once the data file is closed no further IDS processing may take place.

2.3.1.3 Entry DIRECT. Any individual data record may be retrieved by this entry point through its reference code. Prior to the call to DIRECT, the calling routine must store the binary reference code (page number * 64 + line number) in word 1 of common block C10. Following the call, the appropriate attributes in common C30 will have been set to the values of the desired record.

2.3.1.4 Entry NEXTTT. This entry point retrieves the next record of a specified chain which is identified through the argument. Following the call, the appropriate attributes in C30 will have been set to the values of the record retrieved. An end-of-chain condition is met if the master of the chain has been retrieved. The user must check word 4 of common C10 against the master's record type number for end condition.

2.3.1.5 Entry HEAD. This entry point retrieves the master record of a chain as specified within the calling argument. Following the call, the appropriate attributes in common C30 will have been set to the values of the record retrieved. Also, after this call, all chaining (that is calls to NEXTTT) will begin with the first record within the chain.

2.3.1.6 Entry STORE. By specifying the record type's alphabetic name, this entry point creates the record of the type given. Prior to the call the appropriate attributes in common C30 must have been set to the desired values.

2.3.1.7 Entry RETRV. This entry point retrieves primary or CALC records. The record to be retrieved is identified by the record type alphabetic name. Before the call to retrieve a primary record, common C15 must contain the BCD representation of the desired record reference code. Before the call to retrieve a CALC record, attribute DESIG must be set. Following the call, the appropriate attributes in common C30 will have been set to the value of the record retrieved.

2.3.1.8 Entry MODFY. This entry point modifies the current record of the type specified through the argument. Before calling MODFY, the record to be modified must be the current record of that type. Also, all attributes which are to be modified must be set to the new values. In addition, be aware that if a match key is modified, the current record will be moved to a new chain based on the new match key value.

2.3.1.9 Entry DELETE. This entry point deletes the current record of the type specified through the argument. Before calling DELETE, the record to be deleted must be the current record of that type.

2.3.1.10 Entry CLEANP. This entry point assures that all updated IDS pages which are currently in buffers are written to permanent storage. It is used primarily before an IDS save tape is created.

2.3.1.11 Error Recovery. When an error occurs during IDS processing, an error code is inserted in word 6 of common block C10 and subroutine ERPROC is called (see section 3). Calling routines may control the action which ERPROC will take by altering the entries in common block ERRCOM. The action codes are:

ABORT - Core dump occurs and processing stops. (This is the default for NORMAC)

FLAG - Processing continues for this sentence but error flag (common block OOPS) is set. (This is the default for CHCKAC)

PASS - Processing continues -- no action is taken.

Either of the action codes (NORMAC or CHCKAC) may be changed. Error codes may be added to the list for CHCKAC (CHEKS, the list, is originally empty). The report code on which the ERPROC message appears may also be changed. If the contents of ERRCOM are altered they are not reset by COP.

2.3.2 Entry Points for Header Creation and Identification. The two entry points in this section enter headers into the data base and allow calling programs to obtain the BCD reference code of a header for use with a call to RETRV. Each entry has the same four parameters:

- 1 - BCD reference code (Type CHARACTER*8)
- 2 - Value for CLASS attribute
- 3 - Value for SIDE attribute
- 4 - Record Type Name

Note: The first parameter should always be checked after the call as a value of '00000000' indicates an error has occurred.

2.3.2.1 Entry HDPUT. This entry point creates a header and saves its BCD reference code in the data organization index. Either or both values for CLASS or SIDE may be blank.

2.3.2.2. Entry HDFND. This entry point retrieves the BCD reference code of a header for use in a call to RETRV. The BCD code retrieved will be that of a header whose values match those set in parameters 2, 3, and 4. If any of these parameters is blank (including Record Type Name) it is not checked. Note that if both CLASS and Record Type Name are blank the code returned is unpredictable.

(see section 2.3.3.2) or is a pending operation to be prefixed to the next logical instruction. (The usual case here is an AND or OR instruction followed by a NOT instruction.) The instruction codes used with this format appear in table 5.

2.3.3.4.3 Instructions Using Internal Variable. Each of these instructions consists of an instruction code followed by the index of an internal variable. The instruction codes used with this format appear in table 6. An internal variable is the storage of intermediate mathematical calculations. The instruction code informs routines where the storage resides.

2.3.3.4.4 The General Instruction Format. The majority of instructions use this format. As shown in table 4 this format has five variations and may contain from three to nine words. The first major branch occurs by checking word 2. If '9' or '10' is found a constant follows. If '6' is found the value is an attribute and must be retrieved. The other major branch occurs with attributes. If word '5' is zero the attribute is not modified. Otherwise, the information necessary to retrieve the proper record containing the desired attribute is given. This occurs as a result of an 'OF' phrase. The instruction codes which use the general instruction format are found in table 7.

2.3.3.5 Input Instructions Code Entry Points. Routines in the QUICK system may access the input instruction array through the entry points: INSGET, INSPUT, INSFLS, and INSDel (see table 1).

2.3.3.5.1 INSGET. This entry point moves a specified portion of the input instruction array into a specified array. The arguments are the local routine's array, the index of the first word to retrieve and the number of words to retrieve.

2.3.3.5.2 INSPUT. This entry point inserts new instructions. The arguments are the local routine's array which contains the new instructions, an index which is set to the word to be filled first minus one, and the number of words to be inserted. Note that the second argument is returned with the index number of the last word filled.

2.3.3.5.3 INSFLS. This entry point assures that all updates to the instruction array are recorded. It should be called after a completed series of calls to INSPUT. It has no arguments.

2.3.3.5.4 INSDel. This entry point deletes the input instruction table. It has no arguments.

2.3.3.6 Instruction Code Example. Through text English commands, the user has at hand a powerful method of input data generalization. The series of instruction codes which is simply a translation of the input commands, then, is likewise non-restrictive. The previous subsection outlined 11 of the various combinations that instruction codes could have. In

order to summarize and draw together many of the pertinent points concerning instruction code construction, table 8 is given. This example command presents a sentence of a complex nature as well as the instruction code array that would be constructed by the COP. Note that by logically scanning the array (calls to INSGET) modules may readily compute the desired result.

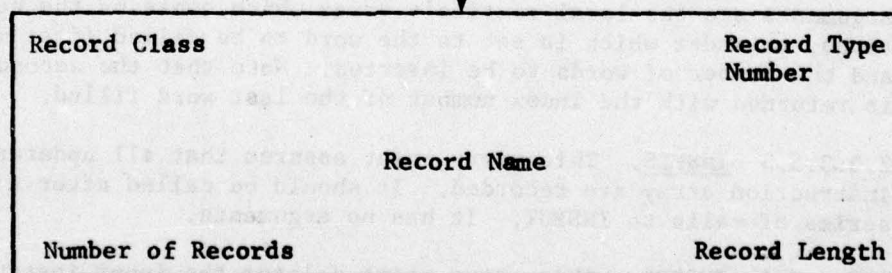
2.3.4 Entry Points for Input Line Processing. The three entry points in this section control the printing of the lines of text English input by the user. The normal call (INPRIN) saves the current line and prints the previous line. The LGPRIN entry is similar to INPRIN except that along with the current line it saves an asterisk to show that the current line was read as part of a long string. Finally, the ERPRIN entry is called to print the current line prior to any error message concerning it.

2.4 QUICK's Data Base Structure

The structured data base is centralized onto one data file. User-directed or module-generated data is properly included within the data base structure automatically by the IDS system as directed through the COP.

Modules as they are executed do not generate new data files linkage. All linkage must be initially defined prior to execution. Only the contents of data records are updated, not the structure. Figures to be presented within this subsection define the structure.

The IDS data records, as given in future subsections, are depicted with a rectangular block and has a format as shown below.



The entries in the block are:

- o Record Class - Indicates how records are stored and accessed. A 'C' defines calculated (CALC) records, a 'P' Primary Records, or an 'S' for Secondary Records
- o Record Type Number - Unique number assigned to the record

Table 8. (Part 5 of 5)

<u>ARRAY NUMBER</u>	<u>ARRAY VALUE</u>	<u>DESCRIPTION</u>
86	0	
87	50*	Load Numeric
88	6	Numeric is an attribute
89	157	Attribute's address (SPDLO)
90	141	Attribute's number (SPDLO)
91	0	No OF phrase
92	18*	Set equal to numeric
93	10	Numeric is a constant
94	900.	Numeric constant
95	2*	End of phrase
96	1*	End of clause

*Instruction code.

- o Record Name - COBOL name for the record
- o Number of Records - Estimated number of data records denoted by the record block
- o Record Length - Number of characters requested for the block
- o Double Line - Denotes CALC records

Individual record blocks are connected with vectors and arrows representing each chain relationship in the file. Beside each chain vector, the chain name is entered. Below each chain name an indication is given to describe how records are positioned or entered in the chain. The appropriate entries are: 'F' for First, 'L' for Last, 'S' for Sorted, 'B' for Before, and 'A' for After.

Organizationally, the QUICK integrated data base may be divided into two parts: the scenario (or gaming) data and the organization data.

2.4.1 Scenario Data Structure. Figure 6 is a picture of the scenario data structure. However, as this structure is quite complex, it will be divided into three parts for discussion purposes. The figures given for the subdivisions are incomplete in that they do not have connected to them the chains which interrelate the three subdivisions.

2.4.1.1 Target Data Structure. Figure 7 shows the target data organization. The TARGET record is the central record type of the data base and is a CALC record. The principal hierarchy is that of target class header (TGTHD) target type (TARGTY) and individual target (TARGET). Targets are grouped by region and complex. The TARGXX chain links some individual target records to additional data. In one case the data is that for a recovery base (RECBTG). In the other case the target is also a missile, bomber, or tanker base (MSBMTG).

Figure 7 also shows refuel points (REFPNT) associated with their region.

2.4.1.2 Weapon Data Structure. Figure 8 shows the weapon data structure. One hierarchy contains the weapon class (WEPHD), weapon type (WEAPON), weapon type subdivided by payload (WEPSUB) and the individual weapon base (MSBMTG). There is also a warhead class (WARHD) with warhead type as details (WRHEAD). Weapons are connected to their warheads via a payload table (PAYTBL) which is master of one chain containing all weapon subtypes (WEPSUB) which utilize that table and another chain which contains counts (PYLDCT) of the various warhead types.

Finally, the QUICK system creates weapon groups (WEPNGP) which has a payload table and a number of individual bases (MSBMTG). These groups are also assigned a geographic region.

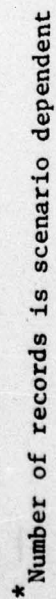
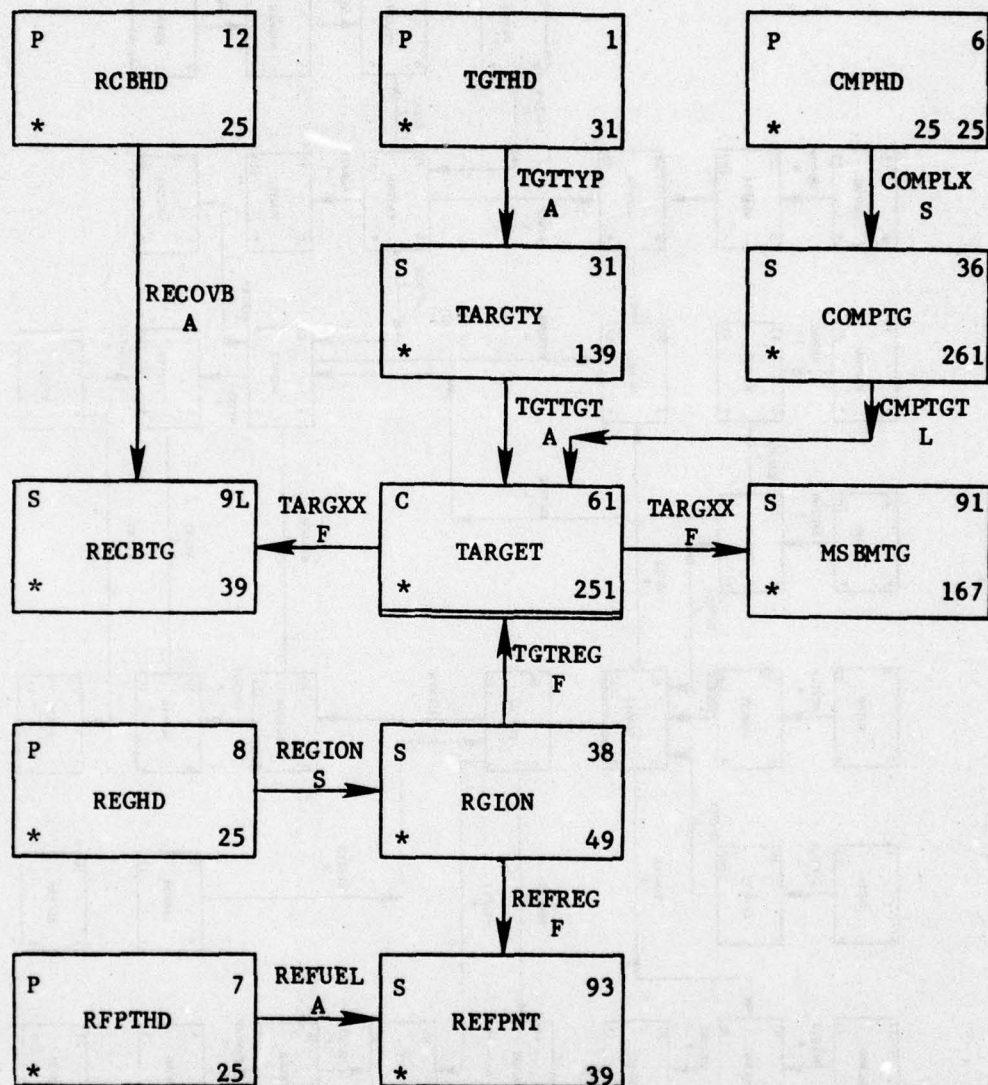
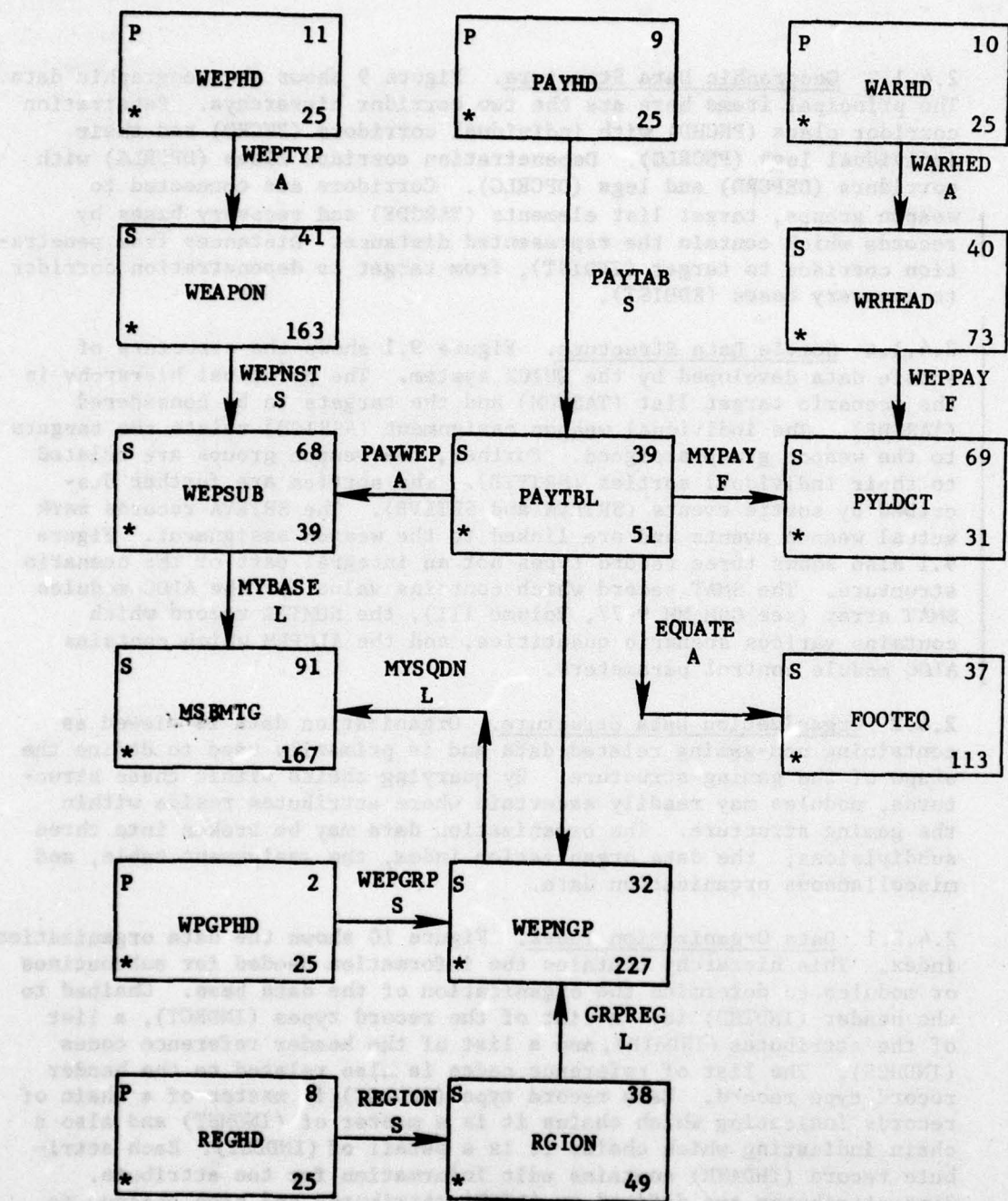


Figure 6. Scenario Data Structure



* Number of records is scenario dependant

Figure 7. Target Data Structure



* Number of records in scenario dependant

Figure 8. Weapon Data Structure

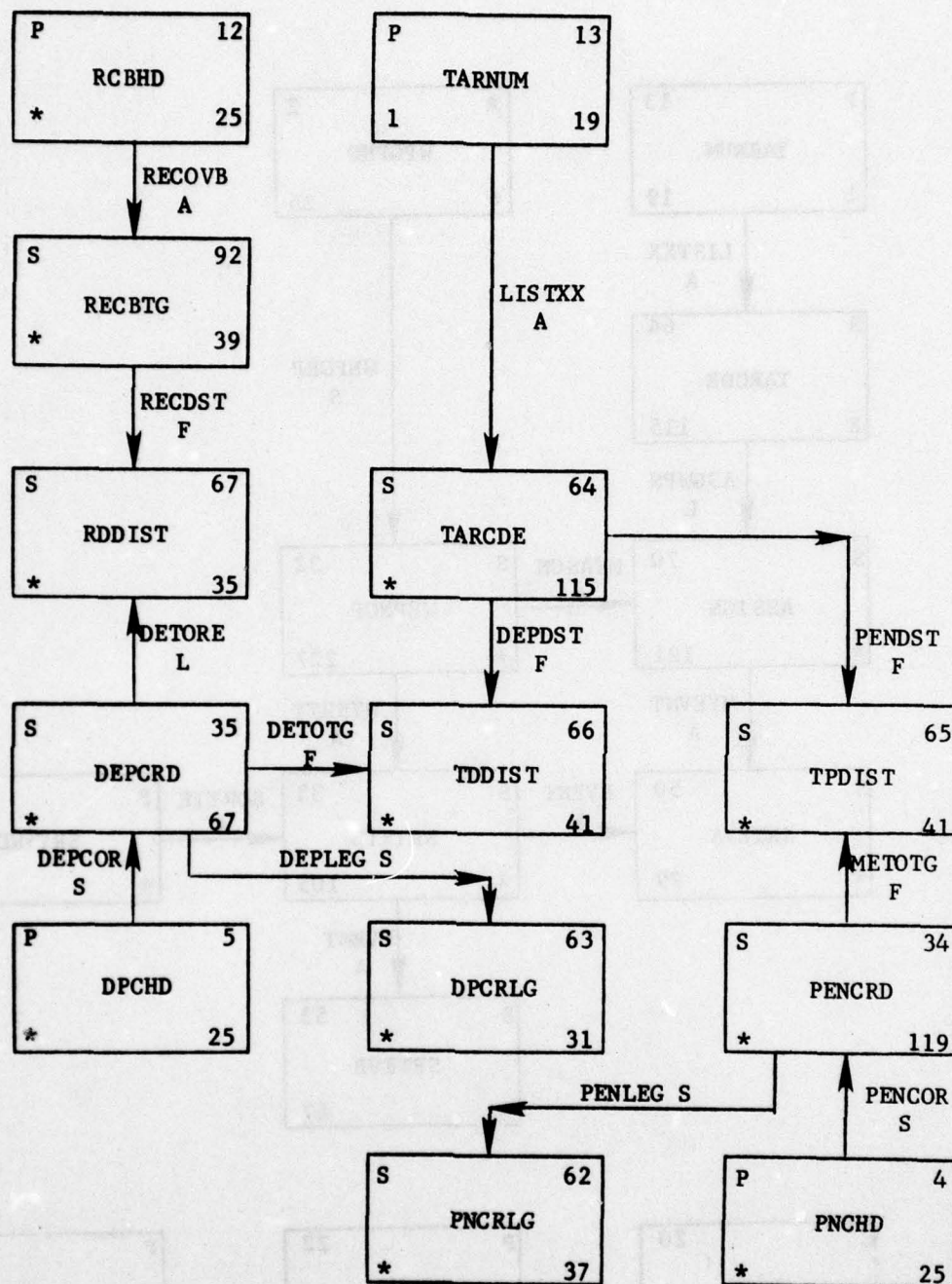
2.4.1.3 Geographic Data Structure. Figure 9 shows the geographic data. The principal items here are the two corridor hierarchys. Penetration corridor class (PNCHD) with individual corridors (PNCRD) and their individual legs (PNCRLG). Depenetration corridor class (DPCRLG) with corridors (DEPCRD) and legs (DPCRLG). Corridors are connected to weapon groups, target list elements (TARCDE) and recovery bases by records which contain the represented distance. Distances from penetration corridor to target (TPDIST), from target to depenetration corridor to recovery bases (RDDIST).

2.4.1.4 Sortie Data Structure. Figure 9.1 shows the structure of sortie data developed by the QUICK system. The principal hierarchy is the scenario target list (TARNUM) and the targets to be considered (TARCDE). The individual weapon assignment (ASSIGN) relate the targets to the weapon group assigned. Further, the weapon groups are related to their individual sorties (SRTYTB). The sorties are further described by sortie events (SRTEVA and SRTEVB). The SRTEVA records mark actual weapon events and are linked to the weapon assignment. Figure 9.1 also shows three record types not an integral part of the scenario structure. The SMAT record which contains value for the ALOC modules SMAT array (see CSM MM 9-77, Volume III), the NUMTBL record which contains various scenario quantities, and the ALCPRM which contains ALOC module control parameters.

2.4.2 Organization Data Structure. Organization data is viewed as containing non-gaming related data and is primarily used to define the shape of the gaming structure. By querying chains within these structures, modules may readily ascertain where attributes reside within the gaming structure. The organization data may be broken into three subdivisions; the data organization index, the assignment table, and miscellaneous organization data.

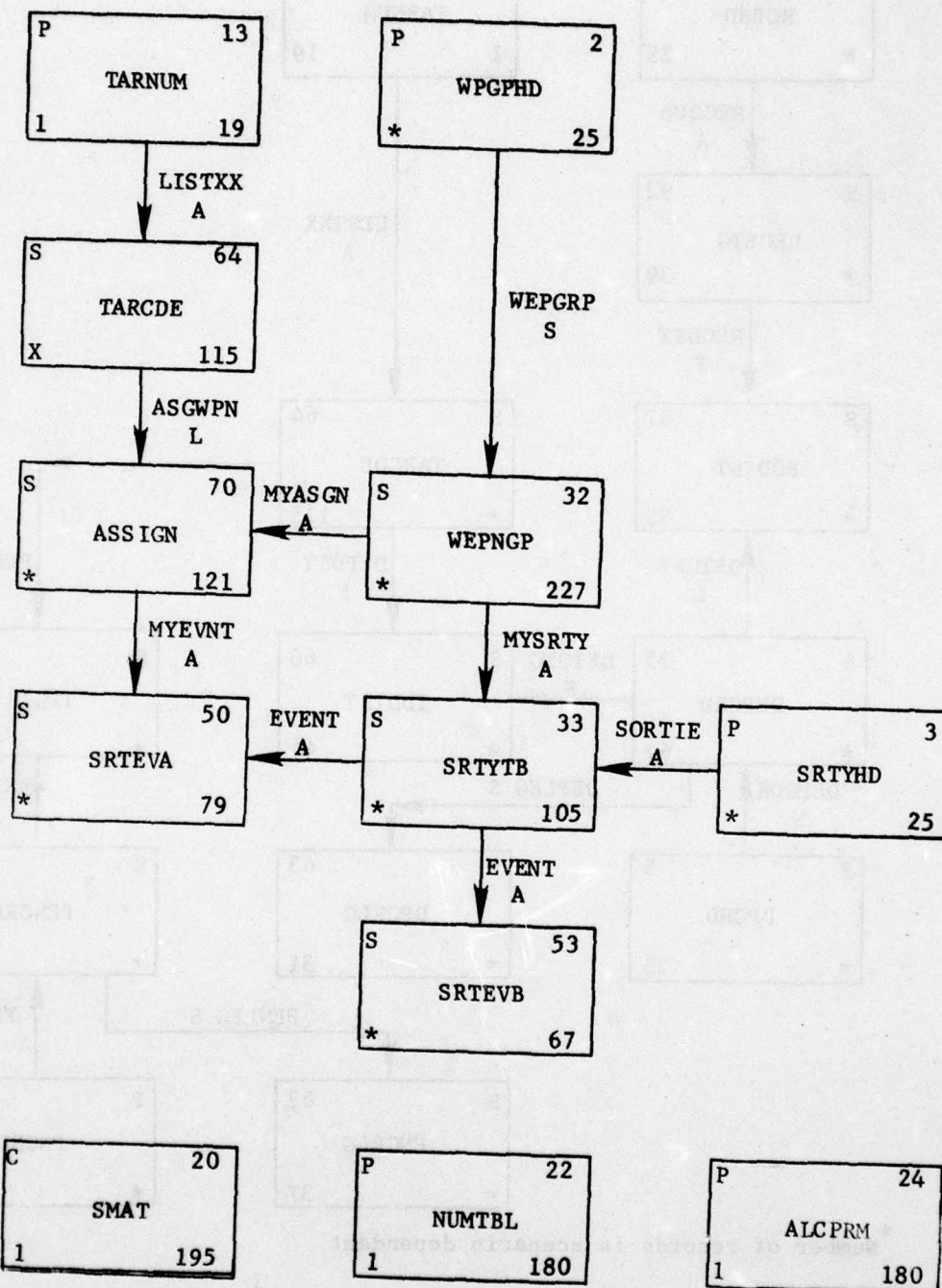
2.4.2.1 Data Organization Index. Figure 10 shows the data organization index. This hierarchy contains the information needed for subroutines or modules to determine the organization of the data base. Chained to the header (INDTHD) is: a list of the record types (INDRCT), a list of the attributes (INDATR), and a list of the header reference codes (INDHCS). The list of reference codes is also related to the header record type record. Each record type (INDRCT) is master of a chain of records indicating which chains it is a master of (INDMST) and also a chain indicating which chains it is a detail of (INDDET). Each attribute record (INDATR) contains edit information for the attribute. Some attributes are defined as 'LIST' attributes and have chained to them records which contain their legal values (ALPHVL). Finally, a record links attributes to the record type(s) which contain them (LINKER).

2.4.2.2 Assignment Table. Figure 11 shows the Assignment Table. This hierarchy contains the information used to determine several attributes for targets based on information from a JAD format record. Chained to the header (ASNTAB) for each side are records containing the valid



* Number of records is scenario dependant

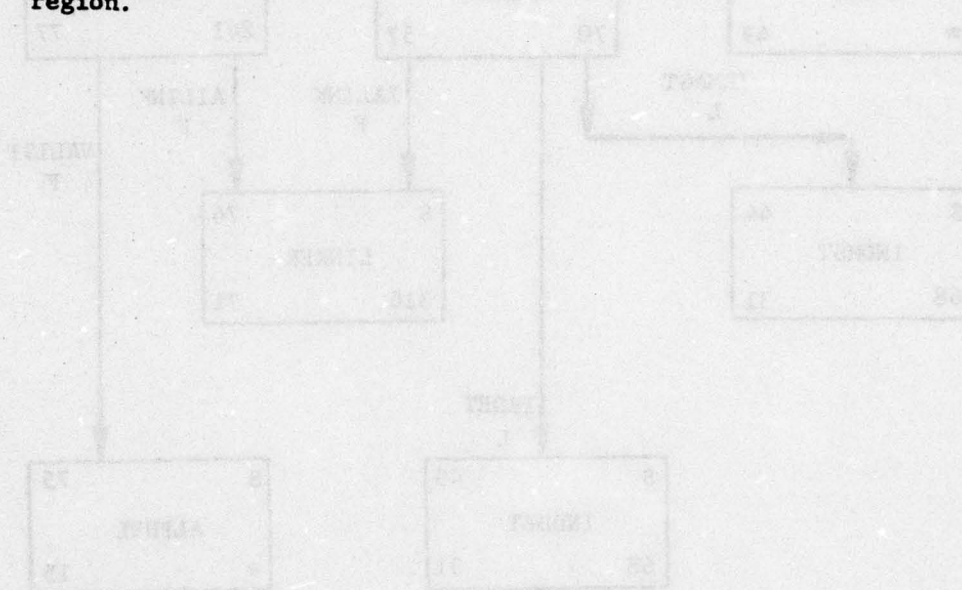
Figure 9. Geographic Data Structure

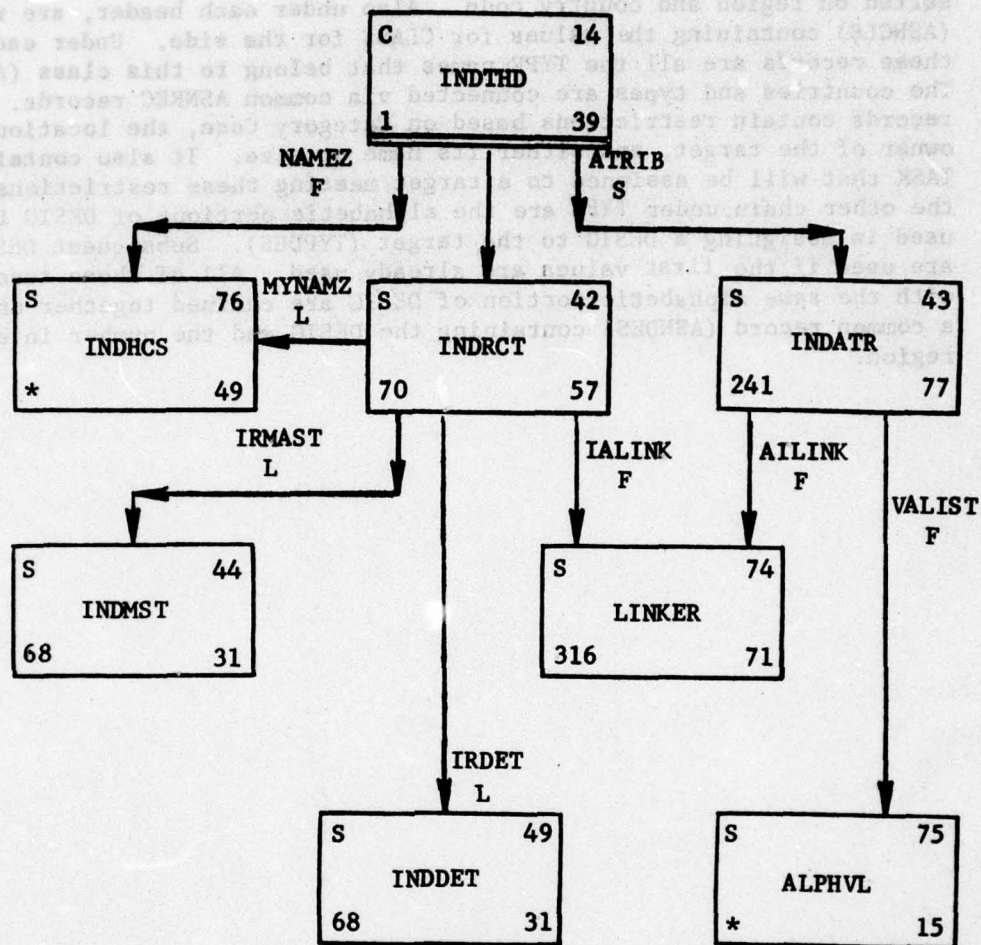


* Number of records is scenario dependant

Figure 9.1 Sortie Data Structure

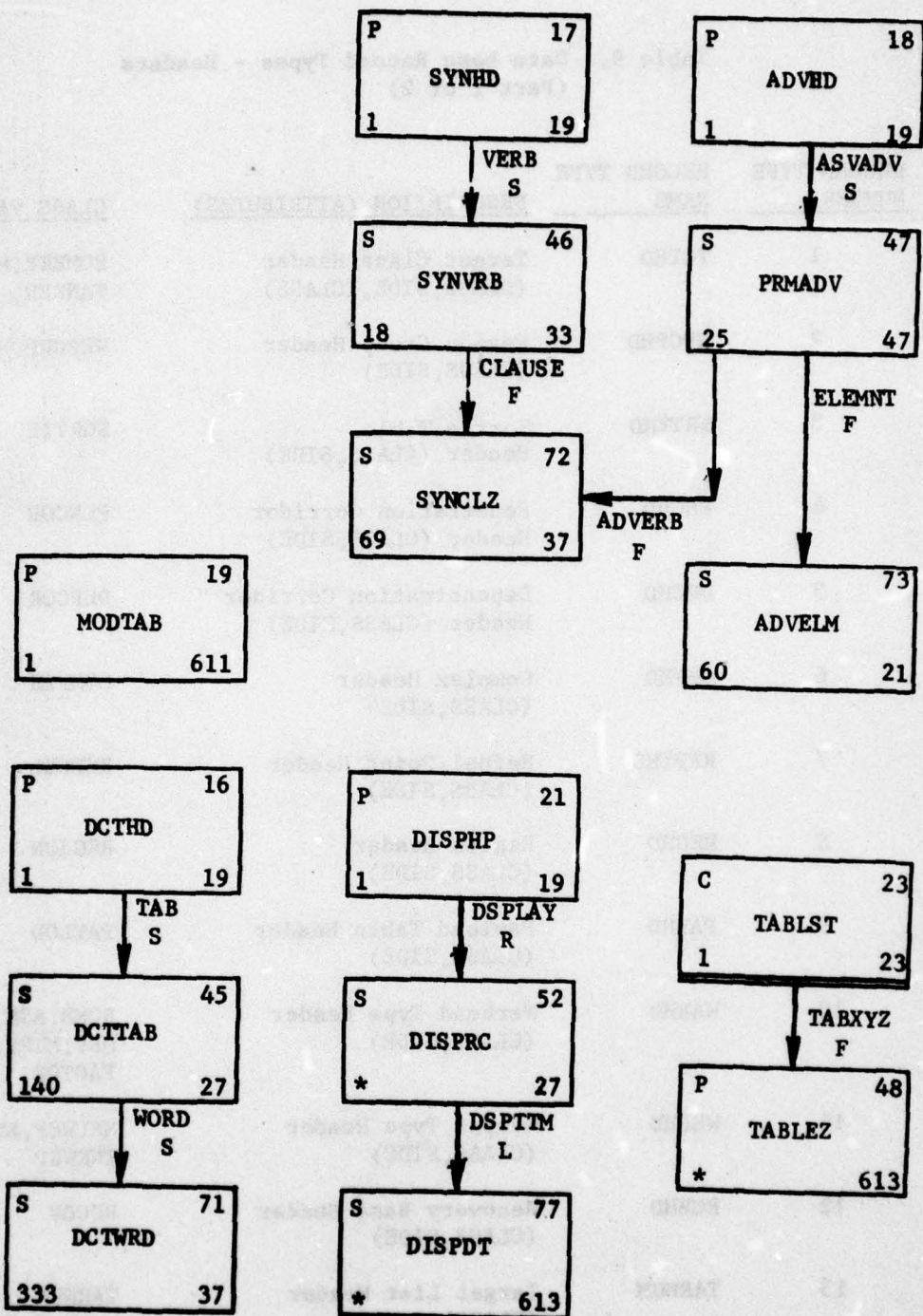
country codes and the region they are in (ASNCTY). These records are sorted on region and country code. Also under each header, are records (ASNCLS) containing the values for CLASS for the side. Under each of these records are all the TYPE names that belong to this class (ASNTYP). The countries and types are connected via common ASNREC records. These records contain restrictions based on Category Code, the location or owner of the target, and either its name or size. It also contains the TASK that will be assigned to a target meeting these restrictions. On the other chain under TYPE are the alphabetic portions of DESIG to be used in assigning a DESIG to the target (TYPDES). Subsequent DESIGs are used if the first values are already used. All of these records with the same alphabetic portion of DESIG are chained together under a common record (ASNDES) containing the DESIG and the number in each region.





* Number of Records is scenario dependent.

Figure 10. Data Organization Index



* Number of records is scenario dependent.

Figure 12. Miscellaneous Organisational Data

**Table 9. Data Base Record Types - Headers
(Part 1 of 2)**

<u>RECORD TYPE NUMBER</u>	<u>RECORD TYPE NAME</u>	<u>DESCRIPTION (ATTRIBUTES)</u>	<u>CLASS VALUES</u>
1	TGTHD	Target Class Header (CLASS, SIDE, ICLASS)	BOMBER, MISSIL, TANKER, etc.
2	WPGPHD	Weapon Group Header (CLASS, SIDE)	WEPGRP
3	SRTYHD	Sortie Table Header (CLASS, SIDE)	SORTIE
4	PNCHD	Penetration Corridor Header (CLASS, SIDE)	PENCOR
5	DPCHD	Depenetration Corridor Header (CLASS, SIDE)	DEPCOR
6	CMPHD	Complex Header (CLASS, SIDE)	COMPLX
7	RFPTH	Refuel Point Header (CLASS, SIDE)	REFUEL
8	REGHD	Region Header (CLASS, SIDE)	REGION
9	PAYHD	Payload Table Header (CLASS, SIDE)	PAYLOD
10	WARHD	Warhead Type Header (CLASS, SIDE)	BOMB, ASM, RV, MRV, MIRV, FACTOR
11	WEPHD	Weapon Type Header (CLASS, SIDE)	MSLWEP, BMBWEP TNKWEP
12	RCBHD	Recovery Base Header (CLASS, SIDE)	RECOV
13	TARNUM	Target List Header (CLASS)	TARNUM

Table 9. (Part 2 of 2)

<u>RECORD TYPE NUMBER</u>	<u>RECORD TYPE NAME</u>	<u>DESCRIPTION (ATTRIBUTES)</u>	<u>CLASS VALUES</u>
14	INDTHD	Data Organization Index Header (CLASS) (This is a CALC record)	INDEX
15	ASNTAB	Assignment Table Header (CLASS, SIDE)	ASSIGN
16	DCTHD	Dictionary Header (CLASS)	DICTON
17	SYNHD	Syntax Directory Verb Header (CLASS)	SYNTAX
18	ADVHD	Syntax Directory Adverb Header (CLASS)	ADVERB
19	MODTAB	Module Link Table (CLASS, link Table (100 words))	MODTAB
20	SMAT	SMAT Array (CLASS) (This is a CALC record)	SMAT
21	DISPHD	REPORT Module Display Header (CLASS)	DIPLY
22	NUMTBL	General Number Table (CLASS)	NUMBER
23	TABLST	Utility Table Header (CLASS) (This is a CALC record)	TABLST
24	ALCPRM	ALOC Control Parameters (CLASS)	ALCPRM

Table 10. Data Base Record Types - Secondary Records
(Part 1 of 4)

<u>RECORD NUMBER</u>	<u>TYPE NAME</u>	<u>DESCRIPTION (ATTRIBUTES)</u>
31	TARGETY	Target Type (CNTRYLO, CNTRYOW, FLAG, FVALT1, FVALT2, FVALT3, FVALT4, FVALT5, FVULN1, NHRDCOMP, NTIMCOMP, T1, T2, T3, T4, T5, TYPE, VULN1 VULN2)
32	WEPNGP	Weapon Group (EXPASM, GBASE, GFRASM, GLAT, GLONG, GNWPADS, GNWPNS, GNVEH, GPKNAV, GREFCODE, GREFTIME, GROUP, GSB, GSBREAL, GSTART, GTYPE, GTYPEFC, GYIELD, IALERT, IREG, MAXSAL, NFIXES, NSAL, NSFIX, NUMALOC) (GROUP is used as a Match Key)
33	SRTYTB	Sortie Table (SDELAY, SDEPEN, SIDERNO, SLAT, SLONG, SLOW, SLOW1, SLOW2, SLOW3, SORTNO, SREFUEL, SVEHNUM)
34	PENCRD	Penetration Corridor (ATTRCO, ATTRPRE1, ATTRPRE2, ATTRPRE3, ATTRSU, CORNUM, DEFDIST1, DEFDIST2, DEFDIST3, DEFRAN, HILOAT, KORSTY, NPRCRDEF, ORLAT, ORLONG)
35	DEPCRD	Depenetration Corridor (CORNUM, MYRECOV1, MYRECOV2, MYRECOV3, MYRECOV4)
36	COMPTG	Complex Target (CNTRYLO, CNTRYOW, DESIG, FLAG, FVALT1, FVALT2, FVALT3, FVALT4, FVALT5, FVULN1, HAZ2, HGZ, GHZ2, ICOMPL, IDHOB, INDEXNO, LAT, LONG, MAXFRA, MAXKILL, MINKILL, MISDEF, NAME, NHRDCOMP, NTIMCOMP, NTINT, RADIUS, TARDEFHI, TARDEFLO, TASK, TGTMULT, TGTNUMB, T1, T2, T3, T4, T5, VALUE, VOZ) (ICOMPL is a match-key)
37	FOOTEQ	Footprint Equation (Contains one hundred words which are module defined)
38	RGION	Region (IREG, CCREL) (IREG is a match key)
39	PAYTBL	Payload table (PAYTBLNM)
40	WRHEAD	Warhead (CEPASM, FFRAC, NAREADEC, NCMS, NDECOYS, NWADS, PAYALT, PDUD, RANGEASM, RELASM, SPEEDASM, TYPE, YIELD)

Table 10. (Part 2 of 4)

<u>RECORD NUMBER</u>	<u>TYPE NAME</u>	<u>DESCRIPTION (ATTRIBUTES)</u>
41	WEAPON	Weapon type (ACTIVE, ALTDLY, BALC, CEP, CMISS, FUNCTI, IPENMO, IRECMO, IREP, LCHINT, MAXSAL, NALTDLY, NMPSIT, PDES, PFPF, PINC, PLABT, RRABT, RANGE, RANGEDEC, RANGERE, REANG, REL, RNQMIN, SIMLUN, SPDLO, SPEED, TOFMIN, TTOS, TYPE) (TYPE is a match key)
42	INDRCT	Index Record Type Record (contains record type name and number)
43	INDATR	Index Attribute Record (ATTRIBN1, ATTRIBN2, ATTRIBNO, ATTRBTYP, ATTRIBAD, ATDEFAULT, ATTRNGHI, ATTRNGLO) (ATTRIBN1 and ATTRIBN2 are match keys)
44	INDMST	Index Master Record (CHAINNAM, MASDETNM, MASDETNO)
45	DCTTAB	Dictionary Tab-character (TABCHAR) (TABCHAR is a match key)
46	SYNVRB	Syntax Verb (CLAUSESW, VERBVAL) (VERBVAL is a match key)
47	PRMADV	Syntax Adverb (ADVERBVL, CLAUSETY, PHRASETY) (ADVERBVL is a match key)
48	TABLEZ	Utility Table (contains 100 words which are module defined)
49	INDDET	Index Detail Record (CHAINNAM, MASDETNM, MASDETNO)
50	SRTEVA	Sortie Event Type A (LAT, LONG, SCUMSURV, SCHANG, SDAMEXP, ADELTIME, SEVCODE, SLOCATTR, SPLACE)
51	ASNCLS	Assignment Table Class (AClass)
52	DISPRC	Display Record (IDISPNAME1, DISPNAME2)
53	SRTEVB	Sortie Event Type B (Same attributes as record number 50)
61	TARGET	Target (BENO, CATCODE, DESIG, HAZ, HAZ2, HGZ, HGZ2, ICOMPL, IDHOB, IGIW, INDEXNO, IREG, ISITE, LAT, LONG, MAJOR, MAXFRA, MAXKILL, MINKILL, MINOR, MISDEF, NAME, NTINT, POP, RADIUS, SIDE, TARDEFHI, TARDEFLO, TASK, TGNUMB, VALUE, VOZ, WACNO) (Record is a CALC record - randomized on DESIG)

Table 10. (Part 3 of 4)

<u>RECORD NUMBER</u>	<u>TYPE NAME</u>	<u>DESCRIPTION (ATTRIBUTES)</u>
62	PNCRLG	Penetration Corridor Leg (ATTRLE, DOGLEG, LAT, LONG)
63	DPCRLG	Depenetration Corridor Leg (DOGLEG, LAT, LONG)
64	TARCDE	Target List Element (TGNUMB, TGTREPCD)
65	TPDIST	Distance from Target to Penetration Corridor (ATTRCD, DISTANCE)
66	TDDIST	Distance from Target to Depenetration Corridor (DISTANCE, DISTDF)
67	RDDIST	Distance from Recovery Base to Depenetration Corridor (DISTANCE)
68	WEPSUB	Weapon Subtype (PAYTBLNM) (PAYTBLNM is a match key)
69	PYLDCT	Count of warhead type in Payload Table (NUMLOAD)
70	ASSIGN	Assignment of Weapon Group to Target (ARRIVE, ASGHOB, DGZLAT, DGZLONG, FIXED, FLMULT, FSALVO, GROUP, KORR, OFFLAT, OFFLONG, PEN, RVAL, TGNUMB)
71	DCTWRD	Dictionary Word (WORDSTR1, WORDSTR2, WORDTY, WORDVL)
72	SYNCLZ	Links Verbs to Adverbs (ADVERBVL, VERBVAL)
73	ADVELM	Gives Legal Elements for elemental Adverbs (ELEMNTTY, ELEMNTVL)
74	LINKER	Links Attributes to Record Types (ATTRIBN1, ATTRIBN2, ATTRIBAD, ATTRBTYP, contains 2 words which are not attributes)
75	ALPHVL	Legal Values for 'LIST' Attributes (ALPLSTVL)
76	INDHCS	Stores Header reference codes (contains header reference code (two words), and the associated CLASS and SIZE values)
77	DISPDT	Display table list (contains 100 words which are created by REPORT)

Table 10. (Part 4 of 4)

<u>RECORD NUMBER</u>	<u>TYPE NAME</u>	<u>DESCRIPTION (ATTRIBUTES)</u>
78	ASNCTY	Assignment Table Country (COUNTRY, REGION)
79	ASNTYP	Assignment Table Type (ATYPE)
80	ASNDES	Assignment Table Desig (DESIGA2, KOUNT1, KOUNT2, KOUNT3, KOUNT4, KOUNT5)
91	MSBMTG	Missile Bomber Target (ADBLI, ADBLR, ALRTDB, ALRTDL, GROUP, IREFUEL, NADBLI, NADBLR, NLRTDB, NLRTDL, NOALER, NOINCO, NOPERSQ, NPRSQ1, NPRSQ2, NPRSQ3, NPRSQ4, NUMDBL, PAYTBLNM, PKNAV, VONBASE, WEPNAME)
92	RECBTG	Recovery base (CAPACITY)
93	REFPNT	Refuel Point (LAT, LONG, IREG)
94	ASNREC	Assignment Table Category (ASNTASK, CATHI, CATLO, CNFLG, MWCAP)
95	TYPDES	Assignment Table Type DESIG (DESIGA2, FULL1, FULL2, FULL3, FULL4, FULL5)

Table 11. Data Base Chains (Part 1 of 4)

<u>CHAIN NAME</u>	<u>MASTER RECORD</u>	<u>DETAIL RECORD</u>	<u>DESCRIPTION</u>
ADVADV	ADVHD	PRMADV	Links Adverb Header to adverbs
ADVERB	PRMADV	SYNCLZ	Links adverb to link to verb
AILINK	INDATR	LINKER	Links Attribute Record to link to record type
ALCLAS	ASNTAB	ASNCLS	Links Assignment table header to assigned classes
ALLDES	ASNTAB	ASNDES	Links Assignment table header to assigned DESIG
ALTDES	ASNTYP	TYPDES	Links Assigned type to link to DESIGs for that type
ALTYPE	ASNCLS	ASNTYP	Links Assigned class to assigned types in that class
ASGWP	TARCDE	ASSIGN	Links target to fix assignments to it
ASNRNG	ASNCTY	ASNREC	Links Assigned country to category ranges for that country
ATRI	INDTHD	INDATR	Links index header to attribute record
CLAUSE	SYNVRB	SYNCLZ	Links verb to link to adverbs
CMPTGT	COMPTG	TARGET	Links complex to targets which make up the complex
COMPLX	CMPHD	COMPTG	Links complex header to complexes
CONTRY	ASNTAB	ASNCTY	Links assignment table header to assigned countries
DEPCOR	DPCHD	DEPCRD	Links depenetration corridor header to depenetration corridors
DEPDST	TARCDE	TDDIST	Links target to distance to depenetration corridors
DEPLEG	DEPCRD	DPCRLG	Links depenetration corridors to its doglegs
DESTYP	ASNDES	TYPDES	Links assigned DESIG to link to assigned type
DETORE	DEPCRD	RDDIST	Links depenetration corridor to distance to recovery base

CH-1

Table 11. (Part 2 of 4)

<u>CHAIN NAME</u>	<u>MASTER RECORD</u>	<u>DETAIL RECORD</u>	<u>DESCRIPTION</u>
DETOTG	DEPCRD	TDDIST	Links depenetration corridor to distance to target
DSPITM	DISPRC	DISPDT	Links display table to its elements
DSPLAY	DISPHD	DISPRC	Links display header to display tables
ELEMNT	PRMADV	ADVLM	Links elemental adverb to its legal elements
EQUATE	PAYTBL	FOOTEQ	Links footprint equations to the proper payload table
EVENT	SRTYTB	SRTEVA	Links sortie table to event type A (weapon event)
EVENT	SRTYTB	SRTEVB	Links sortie table to event type B (non-weapon event)
GRPREG	RGION	WEPNGP	Links region to weapon groups in that region
LALINK	INDRCT	LINKER	Links record type to link to attributes
IRDET	INDRCT	INDDET	Links record type to record showing chains of which it is a detail
IRMAST	INDRCT	INDMST	Links record type to record showing chains of which it is master
LISTXX	TARNUM	TARCODE	Links target list header to elements of the list
METOTG	PENCRD	TPDIST	Links penetration corridor to distance to target
MYASGN	WEPNGP	ASSIGN	Links weapon group to fixed assignments
MYBASE	WEP SUB	MSBMTG	Links weapon sub-type to missile/bomber targets that are its bases
MYEVNT	ASSIGN	SRTEVA	Links weapon assignments to their sortie event
MYNAMZ	INDRCT	INDHCS	Links record type for headers to their reference codes
MYPAY	PAYTBL	PYLDCT	Links payload table to warhead type count
MYSRTY	WEPNGP	SRTYTB	Links weapon group to its sortie tables

Table 11. (Part 3 of 4)

<u>CHAIN NAME</u>	<u>MASTER RECORD</u>	<u>DETAIL RECORD</u>	<u>DESCRIPTION</u>
MYSQDN	WEPNGP	MSBMTG	Links weapon group to missile/bomber targets which provide bases for the group
NAMEZ	INDTHD	INDHCS	Links index header to header reference codes
PAYTAB	PAYHD	PAYTBL	Links payload header to payload tables
PAYWEP	PAYTBL	WEPSUB	Links payload table to weapon sub-type that uses it
PENCOR	PNCHD	PENCRD	Links penetration corridor header to penetration corridors
PENDST	TARCDE	TPDIST	Links target to distance to penetration corridor
PENLEG	PENCRD	PNCRLG	Links penetration corridor to its doglegs
RCTYP	INDTHD	INDRCT	Links index header to record types
RECDST	RECBTG	RDDIST	Links recovery base to distance to depen- tration corridors
RECOVB	RCBHD	RECBTG	Links recovery base header to recovery bases
REFREG	RGION	REFPNT	Links region to refuel points in the region
REFUEL	RFPTHD	REFPNT	Links refuel point header to refuel points
REGION	REGHD	RGION	Links region header to regions
SORTIE	SRTYHD	SRTYTB	Links sortie header to sortie tables
TAB	DCTHD	DCTTAB	Links dictionary header to its tab characters
TABXYZ	TABLST	TABLEZ	Links utility table header to utility tables
TARGXX	TARGET	MSEMTG	Links target to missile/bomber target additional data
TARGXX	TARGET	RECBTG	Links target to recovery base additional data
TGTREG	RGION	TARGET	Links region to targets in region
TGTTGT	TARGET	TARGET	Links target type to targets of that type

Table 11. (Part 4 of 4)

<u>CHAIN NAME</u>	<u>MASTER RECORD</u>	<u>DETAIL RECORD</u>	<u>DESCRIPTION</u>
TGTTYP	TGTHD	TARGETY	Links target header to target types
TYPRNG	ASNTYP	ASNREC	Links assigned type to category range for that type
VALIST	INDATR	ALPHVL	Links 'list' attribute to its legal values
VERB	SYNHD	SYNVRB	Links syntax header to verbs
WARHED	WARHD	WRHEAD	Links warhead header to warhead types
WEPGRP	WPGPHD	WEPNGP	Links weapon group header to weapon groups
WEPNST	WEAPON	WEPSUB	Links weapon type to weapon subtype
WEPPAY	WRHEAD	PYLDCT	Links warhead type to count in payload table
WEPTYP	WEPHD	WEAPON	Links weapon header to weapon types
WORD	DCTTAB	DCTWRD	Links tab character to words with that tab
WPINGP	PAYTBL	WEPNGP	Links payload table to weapon groups using that table

Table 12. Chains Which are Linked to Master

CHAIN NAME	MASTER RECORD	DETAIL RECORD
ADVERB	PRMADV	SYNCLZ
AILINK	INDATR	LINKER
ALTDES	ASNTYP	TYPDES
ASGWPN	TARCD	ASSIGN
ASNRNG	ASNCTY	ASNREC
CMPTGT	COMPTG	TARGET
DEPDST	TARCD	TDDIST
DETORE	DEPCRD	RDDIST
DETOTG	DEPCRD	TDDIST
DESTYP	ASNDES	TYPDES
IALINK	INDRCT	LINKER
METOGP	PENCRD	GPDIST
METOTG	PENCRD	TPDIST
MYASGN	WEPNGP	ASSIGN
MYBASE	WEPSUB	MSEMTG
MYEVNT	ASSIGN	SRTEVA
MYNAMZ	INDRCT	INDHCS
MYSQDN	WEPNGP	MSEMTG
MYSRTY	WEPNGP	SRTYTB
PAYWEP	PAYTBL	WEPSUB
PENDST	TARCD	TPDIST
RECDST	RECBTG	RDDIST
TARGXX	TARGET	MSEMTG
TARGXX	TARGET	RECBTG
TGTREG	RGION	TARGET
TGTTYP	TARGETY	TARGET
TGTTGT	TARGETY	TARGET
WARHED	WARHD	WRHEAD
WEPPAY	WRHEAD	PYLDCT
WPINGP	PAYTBL	WEPNGP

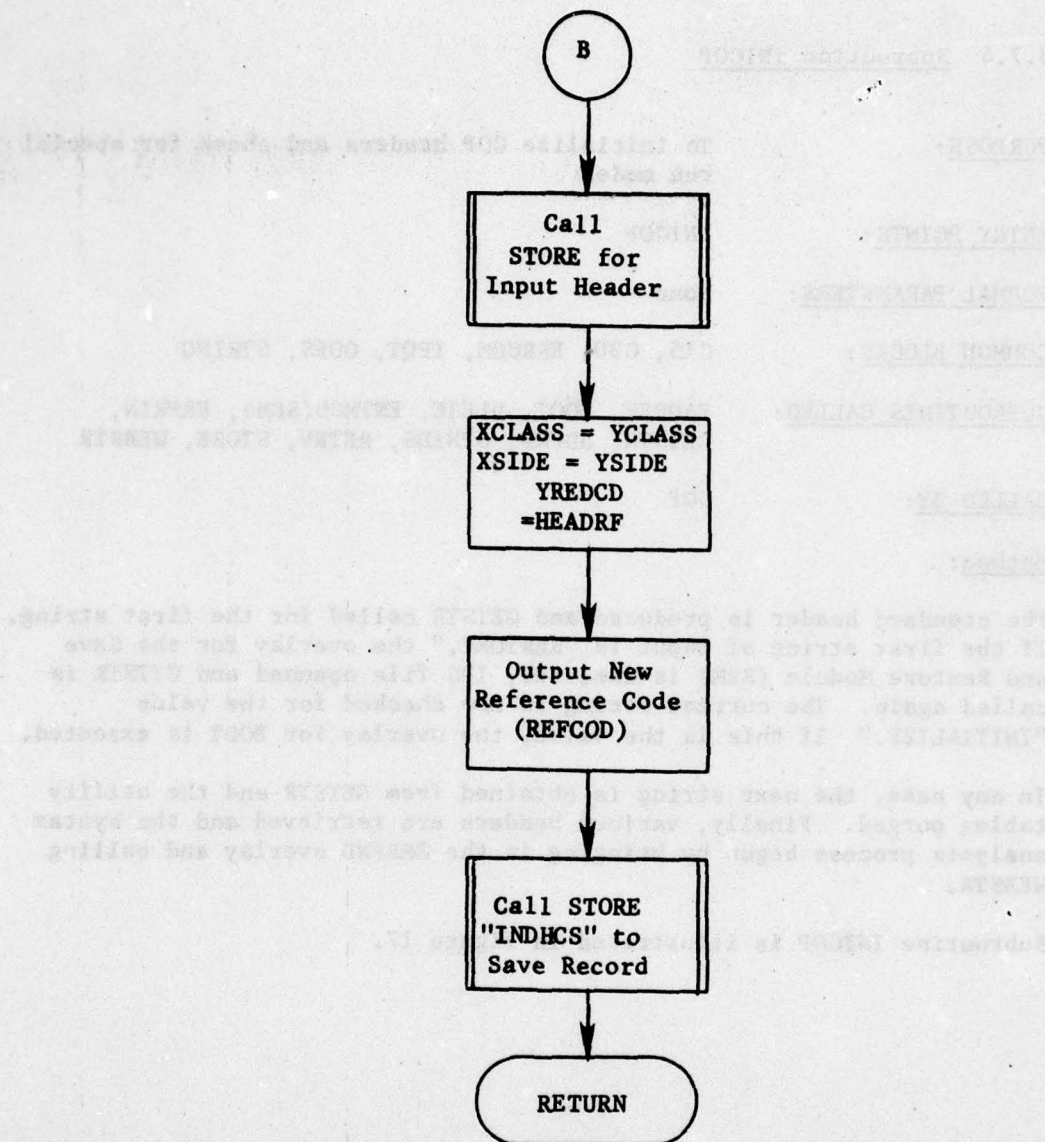


Figure 16. (Part 5 of 5)

3.7.4 Subroutine INICOP

PURPOSE: To initialize COP headers and check for special run modes

ENTRY POINTS: INICOP

FORMAL PARAMETERS: None

COMMON BLOCKS: C15, C30, ERRCOM, IPQT, OOPS, STRING

SUBROUTINES CALLED: BANNER, BOOT, DLETE, ENTMOD(SRM), ERPRIN, GETSTR, HDFND, OPNIDS, RETRV, STORE, WEBSTR

CALLED BY: COP

Method:

The standard header is produced and GETSTR called for the first string. If the first string of input is "RESTORE," the overlay for the Save and Restore Module (SRM) is executed, IDS file opened and GITSTR is called again. The current string is now checked for the value "INITIALIZE." If this is the value, the overlay for BOOT is executed.

In any case, the next string is obtained from GETSTR and the utility tables purged. Finally, various headers are retrieved and the syntax analysis process begun by bringing in the ERRFND overlay and calling WEBSTR.

Subroutine INICOP is illustrated in figure 17.

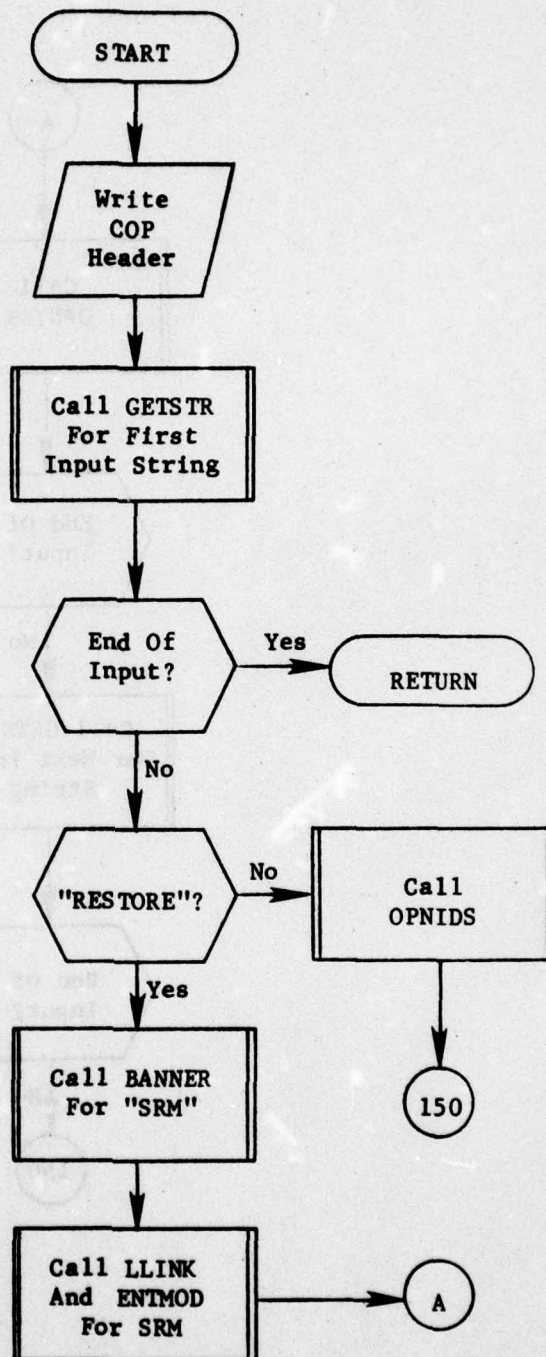


Figure 17. Subroutine INICOP (Part 1 of 4)

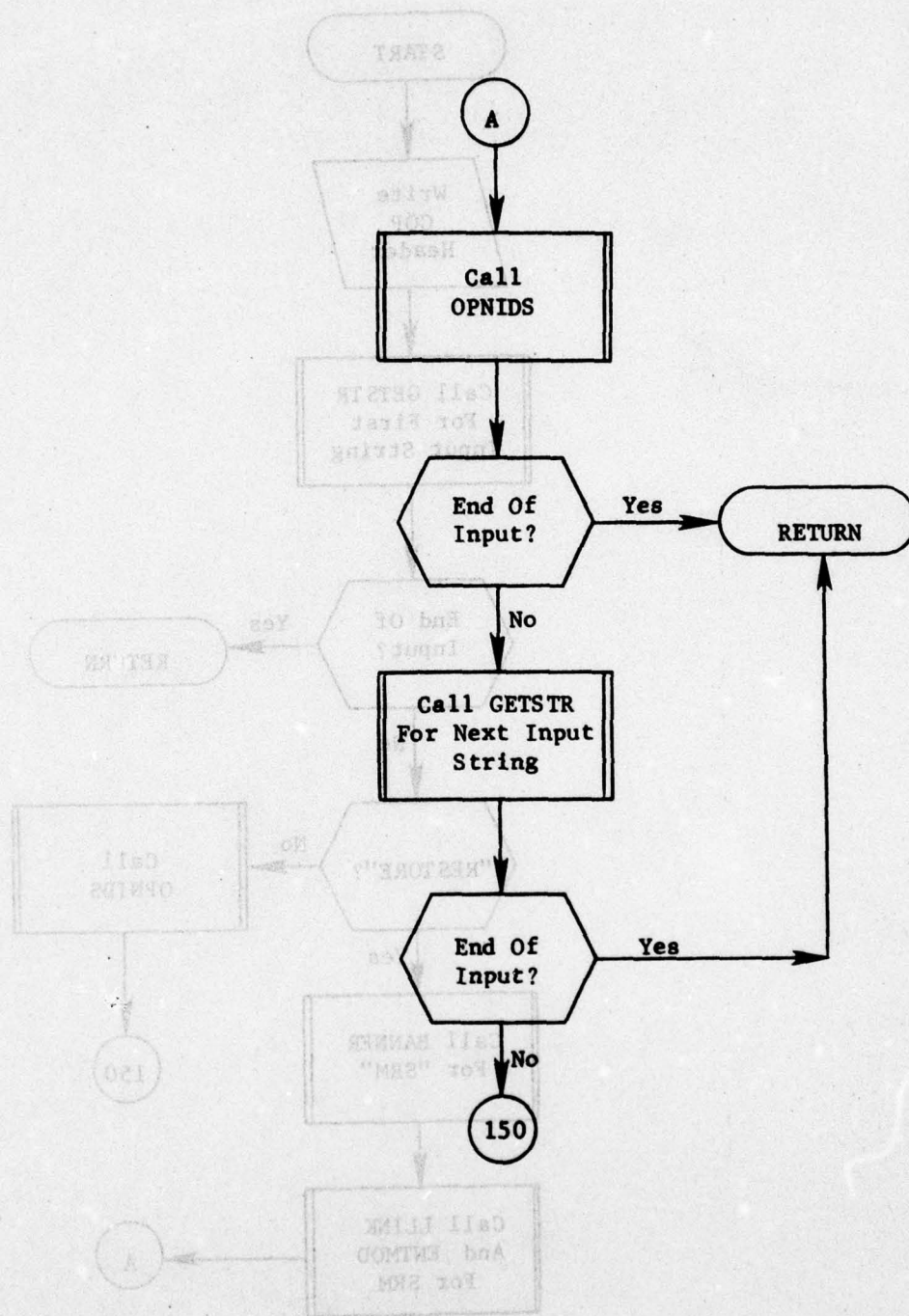


Figure 17. (Part 2 of 4)

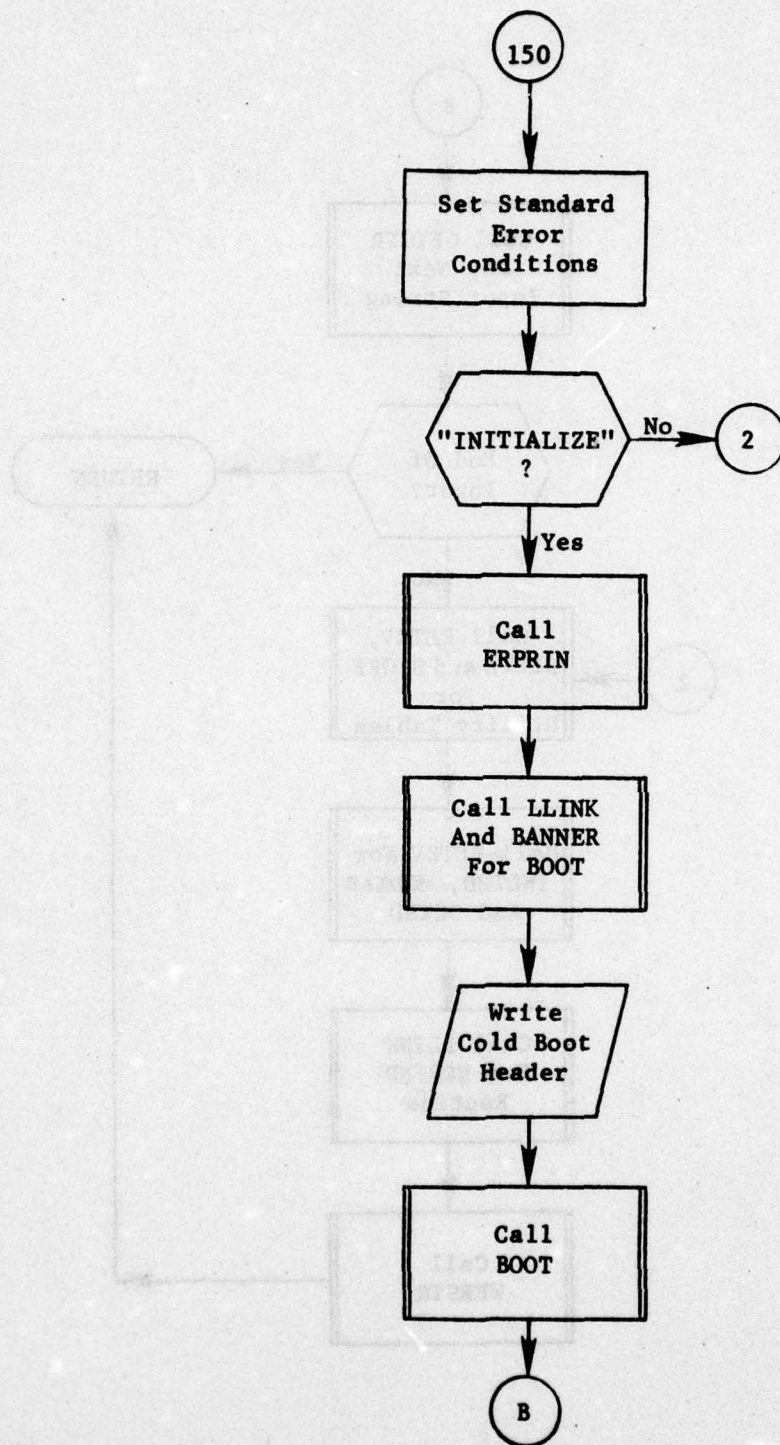


Figure 17. (Part 3 of 4)

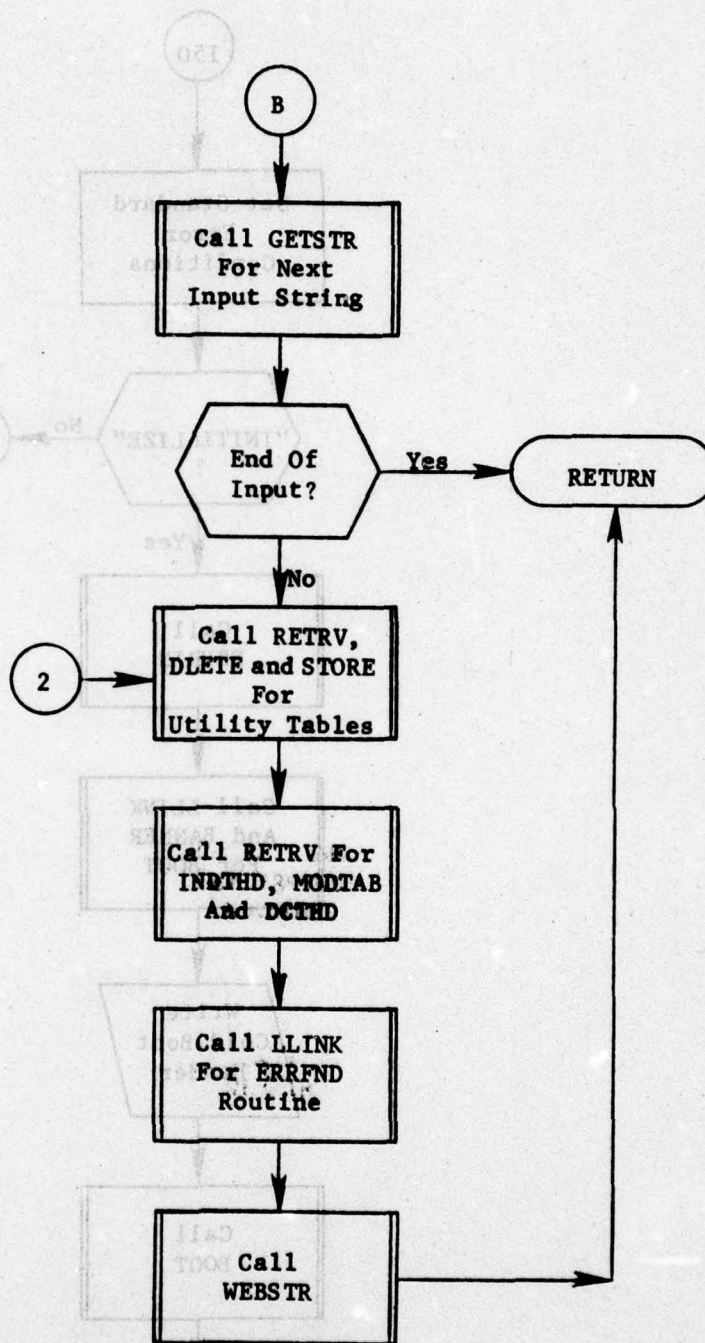


Figure 17. (Part 4 of 4)

3.7.4.1 Subroutine INPRIN

PURPOSE: Control input print

ENTRY POINTS: INPRIN, ERPRIN, LGPRIN

FORMAL PARAMETERS: None

COMMON BLOCKS: IPQT

SUBROUTINES CALLED: None

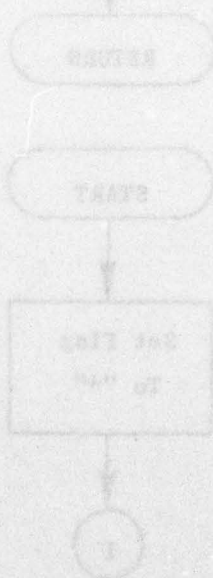
CALLED BY: COP, GETSTR, INICOP, LNGSTR, SYNTAX

Method:

Both the INPRIN and LGPRIN entry points have a similar process. The INPSW switch is checked and if false the containers of HOLD and HFLAG are printed (written to file 11). Then the current input line (INBUF) is stored in HOLD. HFLAG is set according to the entry point - blank for INPRIN, "*" for LGPRIN.

The ERPRIN entry point prints HOLD if it has not been printed and resets INPSW.

Subroutine INPRIN is illustrated in figure 17.1.



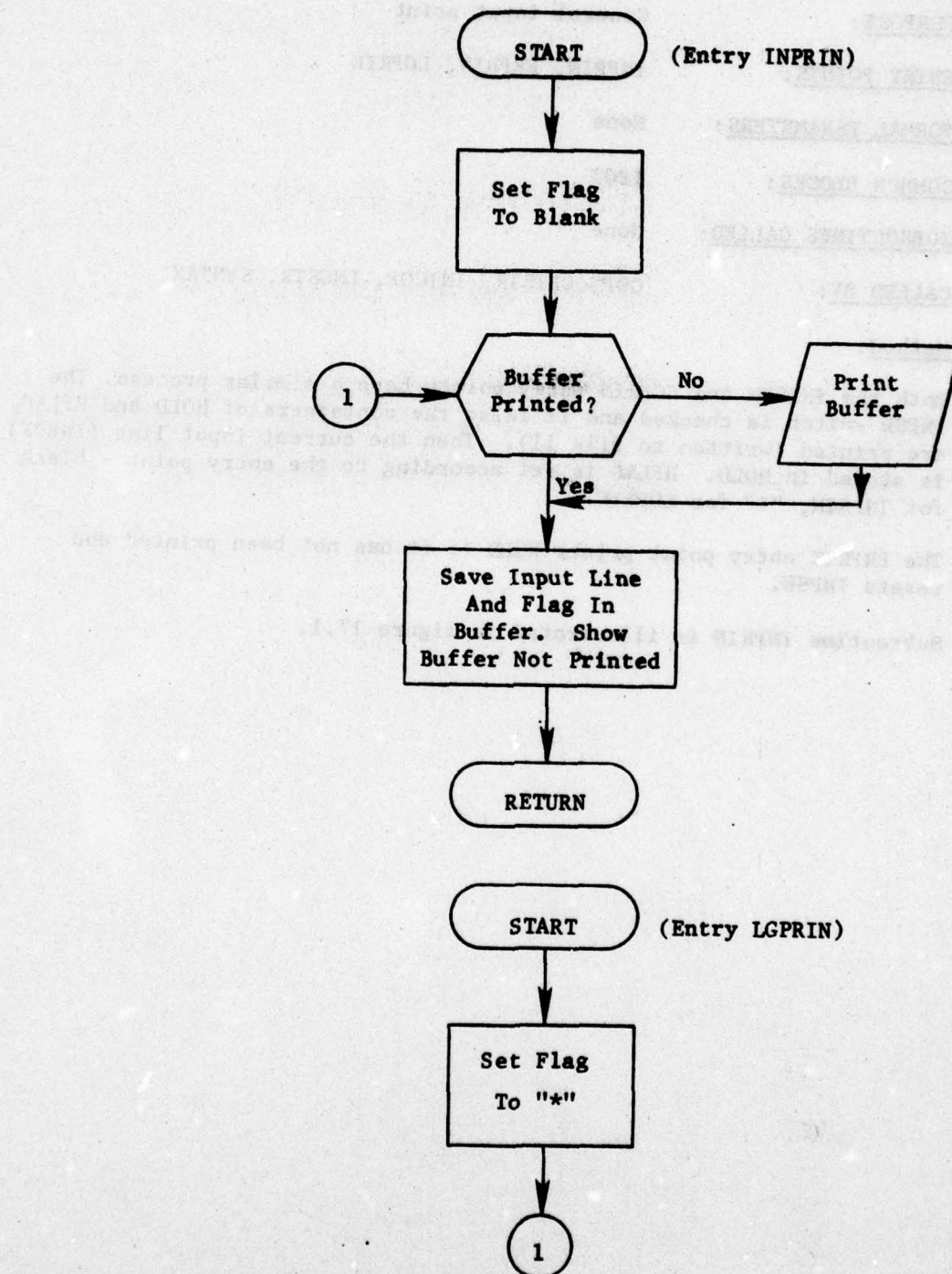


Figure 17.1. Subroutine INPRIN (Part 1 of 2)

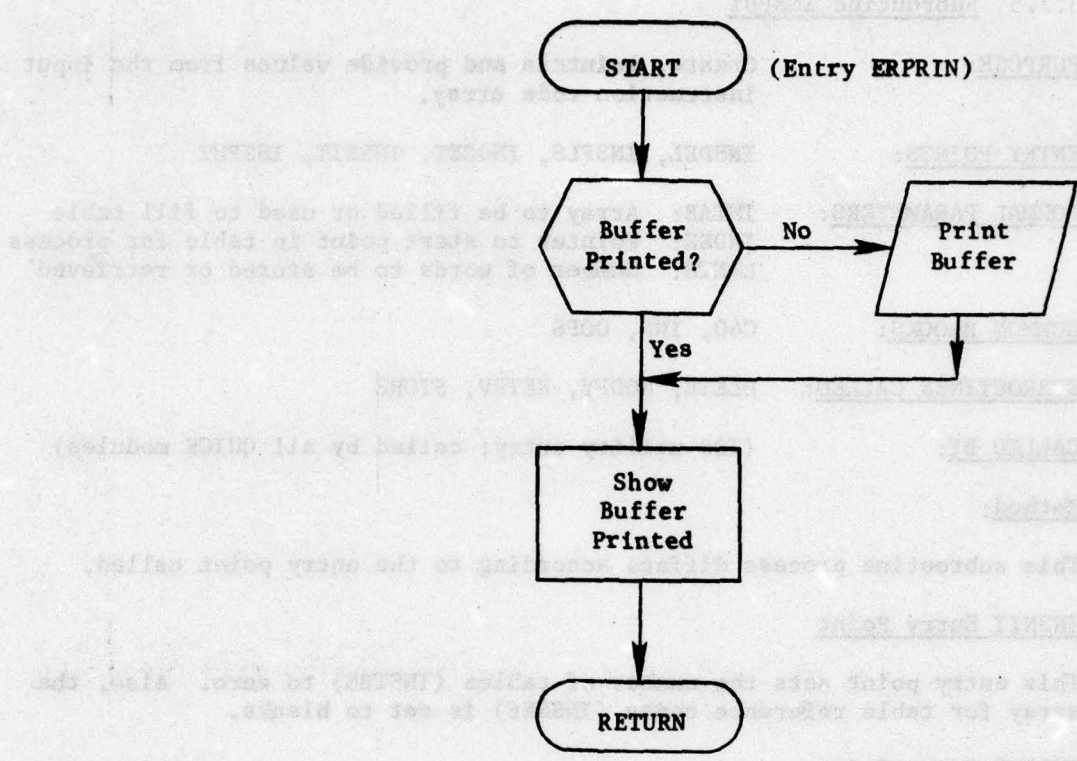


Figure 17.1. (Part 2 of 2)

3.7.5 Subroutine INSPUT

PURPOSE: Create, maintain and provide values from the input instruction code array.

ENTRY POINTS: INSDEL, INSFLS, INSGET, INSINIT, INSPUT

FORMAL PARAMETERS: INTAB: Array to be filled or used to fill table
INDEX: Pointer to start point in table for process
LENTH: Number of words to be stored or retrieved

COMMON BLOCKS: C40, INS, OOPS

SUBROUTINES CALLED: DLETE, MODFY, RETRV, STORE

CALLED BY: (IDS utility entry; called by all QUICK modules)

Method:

This subroutine process differs according to the entry point called.

INSINIT Entry Point

This entry point sets the number of tables (INSTBS) to zero. Also, the array for table reference codes (INSREF) is set to blanks.

INSPUT Entry Point

This entry point enters LENTH elements of INTAB in the Instruction Code array. INDEX is used as a pointer and is incremented prior to each entry. Beginning with the first word of INTAB, INDEX is incremented and the table number and pointer to the word within the table is calculated (each table contains 100 words). If the table is a previously created table, the subroutine assures that the desired TABLEZ table is in C40. Once this is done, the indicated word is set to the new value and MODFY is called. If the desired table is the one being built currently the new value is simply entered. If the desired table is beyond the current table, the current table is moved into the TABLEZ buffer, STORE is called to create a new table and the resulting reference code is saved in INSREF. Finally, the new value is stored in the proper place.

INSFLS Entry Point

This entry point saves the last table in the buffer. The method is to simply set a switch indicating the call was to INSFLS and branch to the appropriate part of the INSPUT code.

3.7.6 Subroutine MODGET

PURPOSE: Execute modules

ENTRY POINTS: MODGET

FORMAL PARAMETERS: None

COMMON BLOCKS: C35, ERRCOM, OOPS

SUBROUTINES CALLED: BANNER, ENTMOD, INSGET

CALLED BY: COP

Method:

First INSGET is called to get the verb's number. This number is used as an index to the module link table (common block C35) to obtain an overlay link name. This name (NEWMOD) is compared to the old name (OLDMOD). If they are different, BANNER is called to display NEWMOD. OLDMOD is set to NEWMOD. System routine LLINK is called to read in overlay NEWMOD and standard module entry point ENTMOD is called. The standard error conditions are now reset. If an error occurred during module execution an error message is produced. Finally, if either the DATA or REPORT module was executed, the error flag is reset to false.

Subroutine MODGET is illustrated in figure 19.

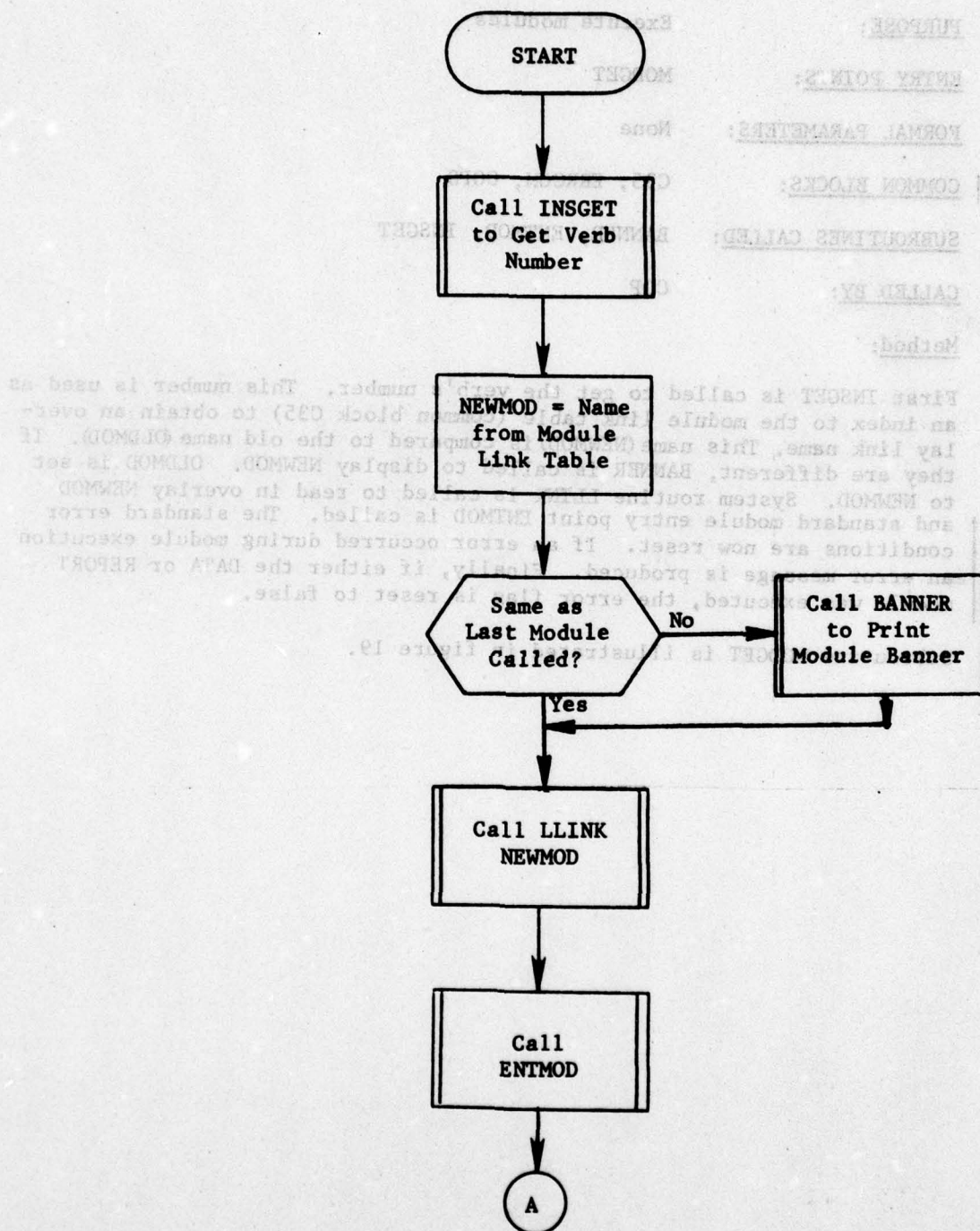


Figure 19. Subroutine MODGET (Part 1 of 2)

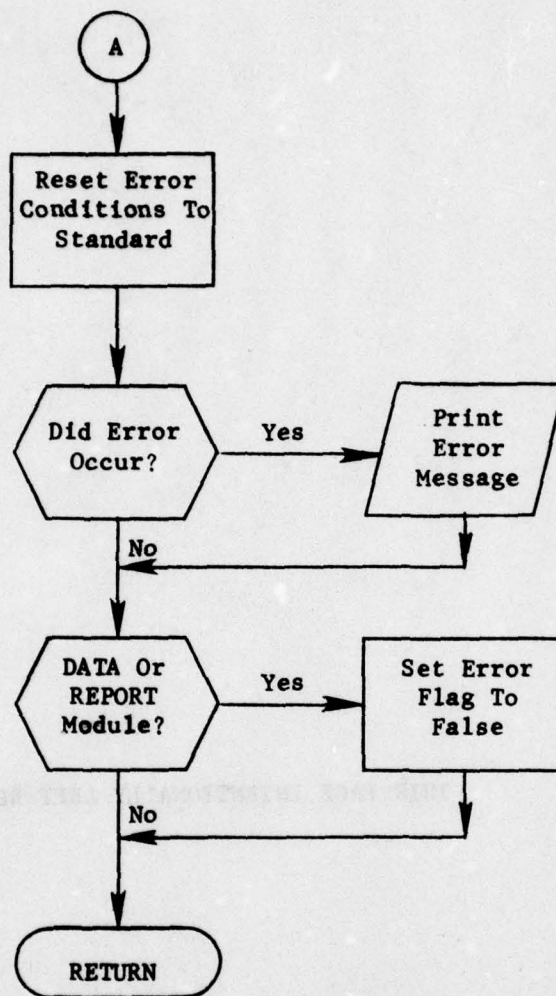
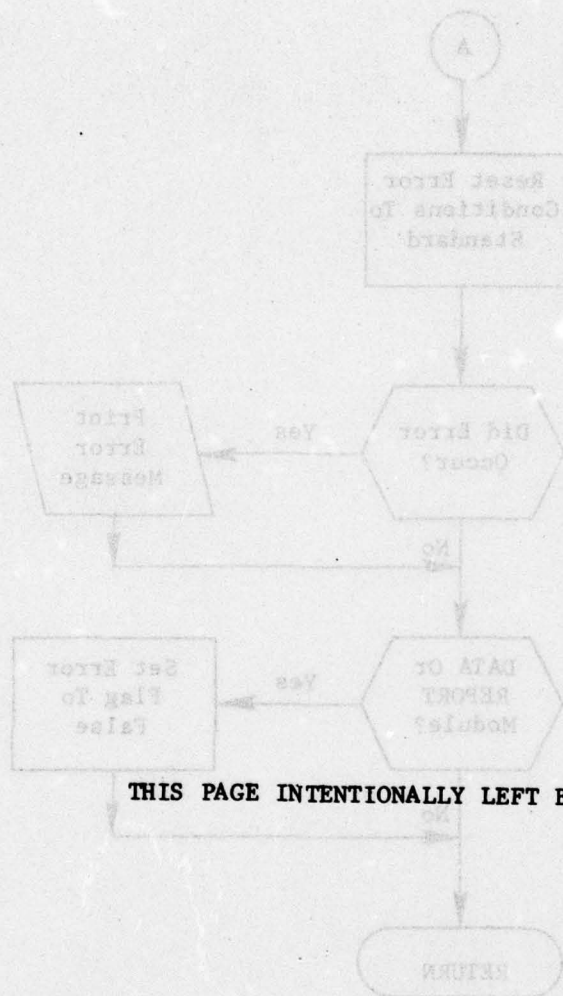


Figure 19. (Part 2 of 2)



THIS PAGE INTENTIONALLY LEFT BLANK

Figure 10. (Part 2 of 2)

3.7.7 Subroutine QDATA

PURPOSE: Performs IDS functions

ENTRY POINTS: CLEANP, CLZIDS, DIRECT, DLETE, HEAD, MODFY, NEXTTT, OPNIDS, RETRV, STORE

FORMAL PARAMETERS: ARGUMENT: Record type or chain name

COMMON BLOCKS: C10, C15, C20, C25, C30, C35, C40, C50

SUBROUTINES CALLED: ERPROC

CALLED BY: (IDS utility entry, called by all QUICK modules)

Method:

The entry points for this routine fall into three groups: those with no argument: CLZIDS, OPNIDS, DIRECT; those whose argument is a record type name: DLETE, MODFY, RETRV, STORE; and those whose argument is a chain name: HEAD, NEXTTT.

CLEANP, CLZIDS, OPNIDS and DIRECT

These entries each carry out a single function. CLEANP dumps altered pages to file storage. CLZIDS closes the IDS file. OPNIDS opens the IDS file for update. DIRECT retrieves the record whose binary reference code is stored in C10.

DLETE, MODFY, RETRV and STORE

For each of these entry points the process is quite similar. The input record type name is looked up in a table of legal record type names (REC-TYPE-TABLE). If it is not in the table the error code is set to '00ORRR'. After the look-up, the subroutine branches to the appropriate code to perform the requested function. In some cases (see notes to figure 20) the requested function is not allowed in which event the error code is set to '000ILC'.

HEAD and NEXTTT

For each of these entry points the process is similar. The input chain name is looked up in a table of legal chain names (CHAIN-NAME-TABLE). If it is not in the table, the error code is set to '00OCCC'. After the look-up, the subroutine branches to the appropriate code to perform the requested function.

After any of the above entry points process have taken place, the error code is checked. If an error has occurred ERPROC is called.

Subroutine QDATA is illustrated in figure 20.

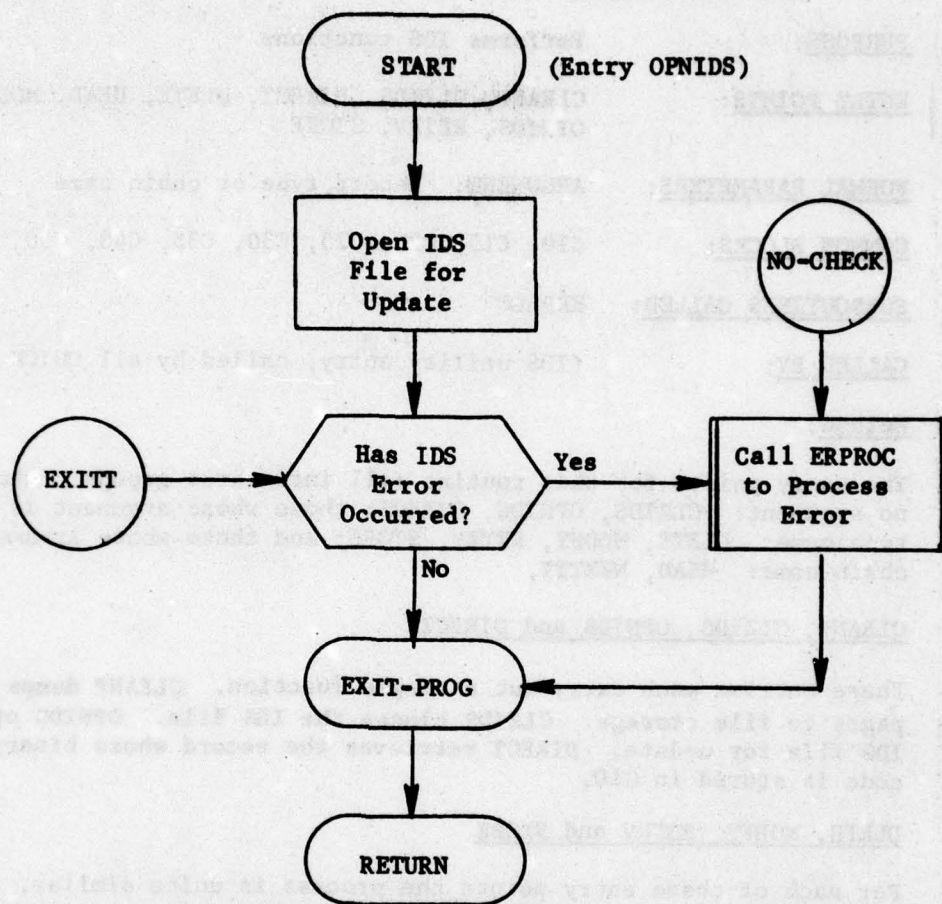


Figure 20. Subroutine QDATA: Entry OPNIDS (Part 1 of 9)

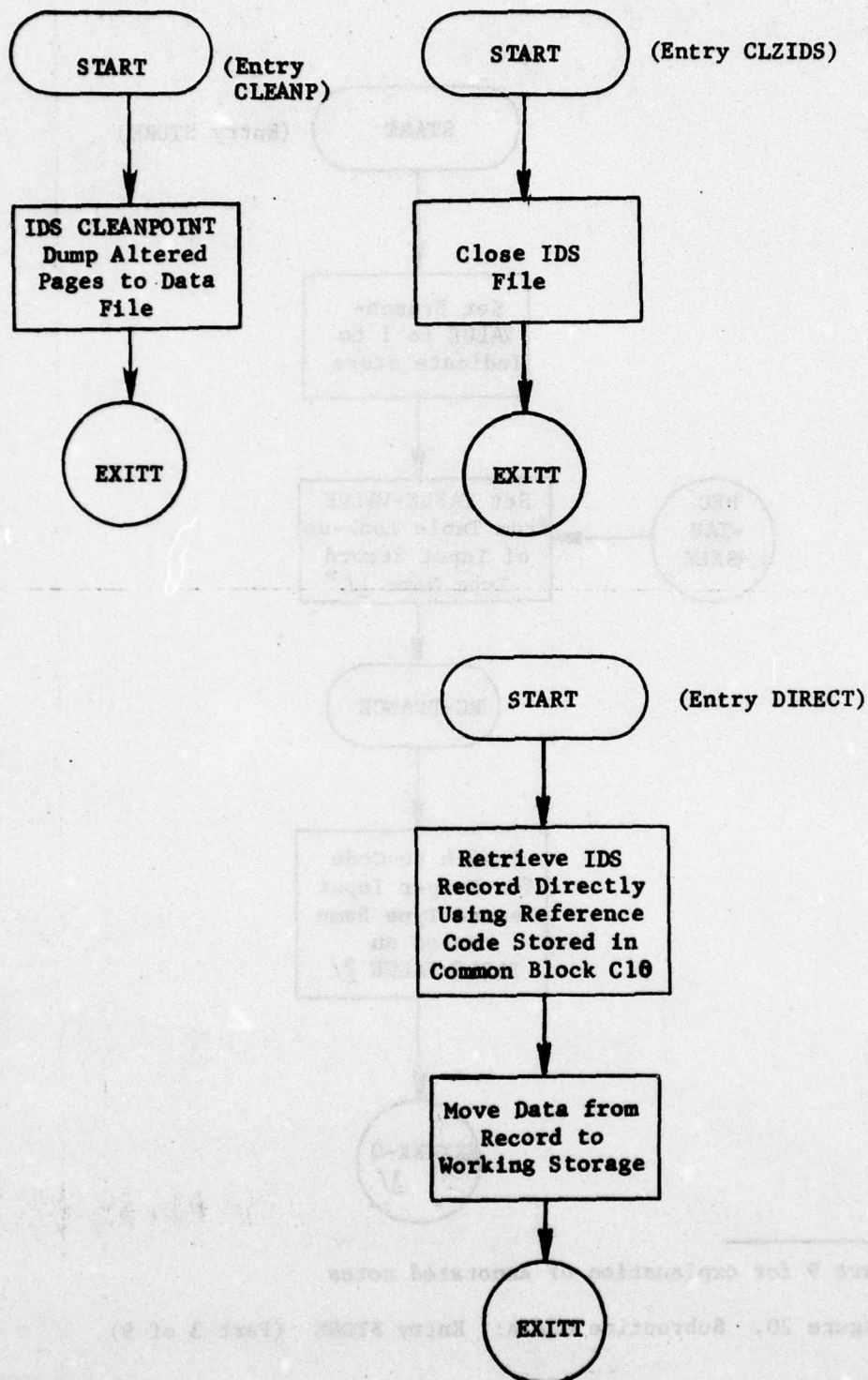
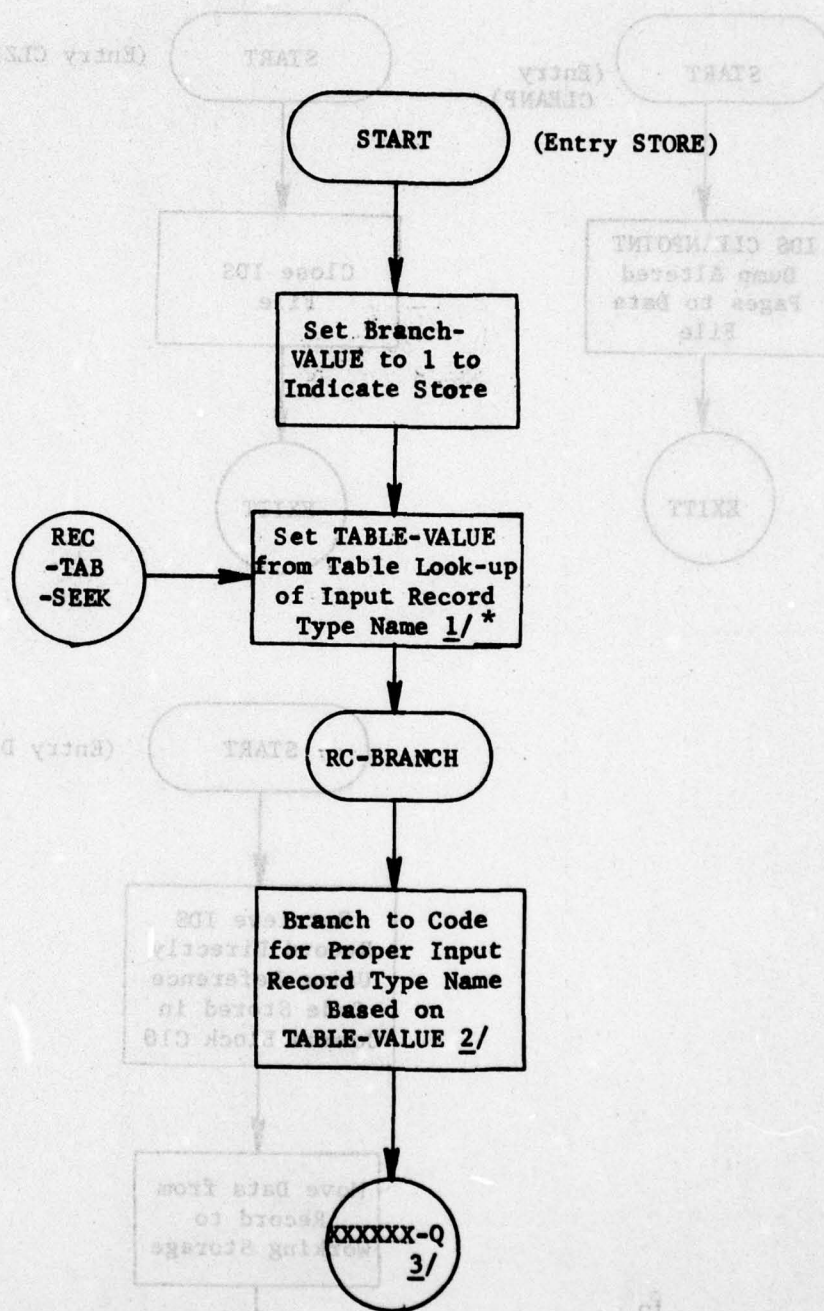


Figure 20. Subroutine ODATA: Entry CLZIDS and DIRECT
(Part 2 of 9)



* See part 9 for explanation of annotated notes

Figure 20. Subroutine QDATA: Entry STORE (Part 3 of 9)

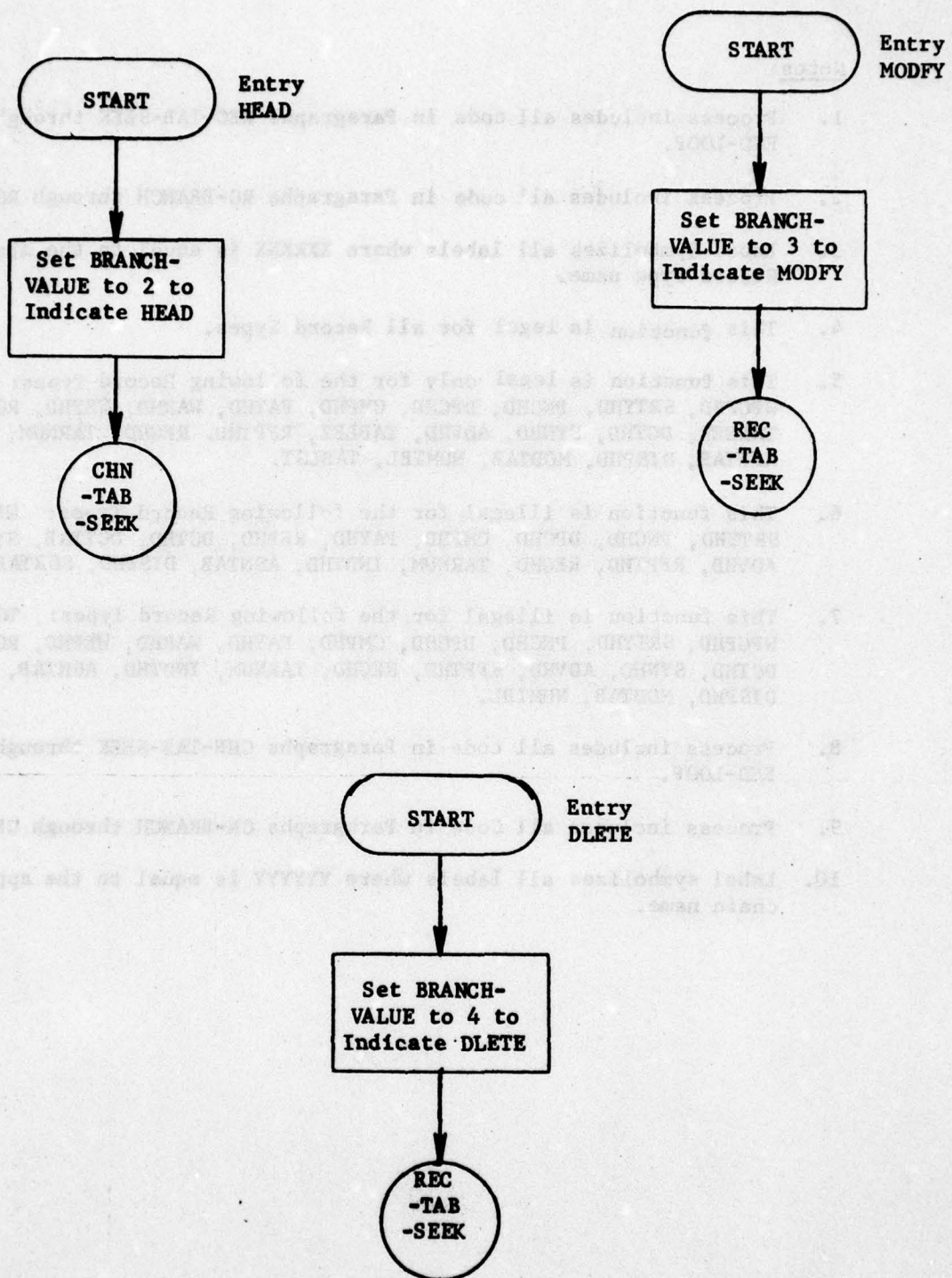


Figure 20. Entries HEAD, MODIFY, and DELETE
(Part 8 of 9)

Notes:

1. Process includes all code in Paragraphs REC-TAB-SEEK through RTS-END-LOOP.
2. Process includes all code in Paragraphs RC-BRANCH through RC-X.
3. Label Symbolizes all labels where XXXXXX is equal to the appropriate Record Type name.
4. This function is legal for all Record Types.
5. This function is legal only for the following Record Types: TGTHD, WPGPHD, SRTYHD, PNCHD, DPCHD, CMPHD, PAYHD, WARHD, WEPHD, RCBHD, TARGET, DCTHD, SYNHD, ADVHD, TABLEZ, RFPTHD, REGHD, TARNUM, INDTHD, ASNTAB, DISPHD, MODTAB, NUMTBL, TABLST.
6. This function is illegal for the following Record Types: WPGPHD, SRTYHD, PNCHD, DPCHD, CMPHD, PAYHD, REBHD, DCTHD, DCTTAB, SYNHD, ADVHD, RFPTHD, REGHD, TARNUM, INDTHD, ASNTAB, DISPHD, SORTAB.
7. This function is illegal for the following Record Types: TGTHD, WPGPHD, SRTYHD, PNCHD, DPCHD, CMPHD, PAYHD, WARHD, WEPHD, RCBHD, DCTHD, SYNHD, ADVHD, RFPTHD, REGHD, TARNUM, INDTHD, ASNTAB, DISPHD, MODTAB, NUMTBL.
8. Process includes all code in Paragraphs CHN-TAB-SEEK through CTS-END-LOOP.
9. Process includes all Code in Paragraphs CN-BRANCH through CNB-F.
10. Label symbolizes all labels where YYYYYY is equal to the appropriate chain name.

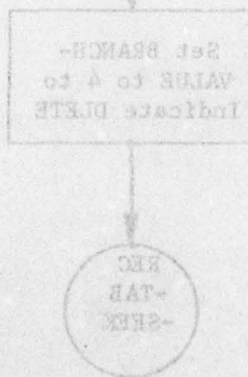


Figure 20. (Part 9 of 9)

3.8 Subroutine BOOT*

PURPOSE: Create and update organizational data

ENTRY POINTS: BOOT

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C20, C30, C35

SUBROUTINES CALLED: DCTFND, HDFND, HDPUT, HEAD, MNMFND, MODFY, NEXTTT, NUMFND, RETRV, RNMFND, SEEKER, STORE, STRMAK

CALLED BY: INICOP

Method:

BOOT reads card images which instruct it as to what actions to take. The card images are in sets which are begun by an introductory adverb and ended with an END card. The first part of the process is to read an adverb (NEWINDEX, RECORDTYP, INDEX, DICTIONARY, SYNTAX, MODULE, HEADER). Each adverb causes the branch IBR to be set to a different value. If an adverb is read which is not recognized processing ceases. The method used for each adverb is different. However, each card image that is read is printed after any action is taken with an appropriate flag (IND).

NEWINDEX

No command cards follow this abverb. The action taken is to set the CLASS attribute and call STORE to create the utility table and index headers.

RECORDTYP

Each card image creates a new INDRCT record. The process is to call NUMFND for the record type number then search the RCTYP chain for a match. If a match is found, IND is set for ignored input. If no match is found, IND is set to show added record and STORE is called to create an INDRCT record.

INDEX

Each card image creates a new record of the type specified in the first field of the card image.

*Main routine of overlay BOOTT.

INDMST or INDDT: RNMFND is called for the record type numbers and to check the validity of both record type names. SEEKER is then called on the appropriate chain (IRMAST or IRDET, respectively) to look for a duplicate. If a duplicate is found the ignored flag (IND) is set. Otherwise, the appropriate attributes are set and STORE is called.

INDATR: First STRMAK is called to obtain the attribute name. Then this name is checked for validity by DCTFND. Next the ATRIB chain is queried by SEEKER. If a match is found IND is set to indicate a change, if not it is set to indicate an add. Now MNMFND is used to get the value of the attribute type. This type is used to determine how to read the default and range fields. Finally, depending upon IND, either STORE or MODIFY is called.

ALPHVL

First STRMAK is called to obtain the attribute name, and the name is validated by DCTFND. Next SEEKER is used to find this attribute on the ATRIB chain. Then SEEKER is used to check the VALIST chain for a duplicate value. Finally, if all decks are correct, STORE is called to add a ALPHVL record.

LINKER

First STRMAK is called to obtain the attribute name and it is validated by DCTFND. Next RNMFND is used to validate the record type name. Next the IALINK chain is searched by SEEKER for a duplicate, which, if found, causes the flag (IND) to be set for a change. Then MNMFND is called to obtain the value of the control mnemonic. Finally, either MODIFY or STORE is called depending upon the flag (IND).

DICTIONARY

Each card image creates a new entry in the dictionary. First STRMAK is called to obtain the input value for the word to be entered in the dictionary. Next DCTFND is called to look for the new word in the dictionary. If the word is found the indicator flag (IND) is set for a change. If neither the word nor its tab character (tab character is formed from the first two characters of the word) is found, STORE is called to create an appropriate tab character record (DCTTAB). Now MNMFND is called to set the word type. If the type is attribute (Type=6) NUMFND is called for the value, and the address and MNMFND is called for the type and identifier flag. These quantities are packed into WORDVL. If not an attribute, NUMFND is called for WORDVL. Finally, either MODIFY or STORE is called.

SYNTAX

Each card image creates a new record of the type specified in the first field of the input card.

SYNVRB: STRMAK is called to obtain the verb and DCTFND is used to validate it. SEEKER is then used to look for a match on the VERB chain and IND set to change if one is found. Next MNMFND is called to obtain the value of the clause switch (ICSW). Finally, either MODIFY or STORE is called.

PRMADV: STRMAK is called to obtain the adverb name and DCTFND is called to validate it. SEEKER is then used to look for a match on the ADVADV chain and the flag set to change if one is found. Next MNMFND is called for the clause type (IXTYP) and the phrase type (ILTYP). Finally, either MODIFY or STORE is called.

ADVELM: STRMAK is called to obtain the adverb name and DCTFND is called to validate it. Next SEEKER is called to find the adverb on the ADVADV chain. MNMFND is now used to find the element's type. If the type is special word (Type=5), or operator (Type=1) DCTFND is used to find the value. Operators are also checked against a list of special characters. Finally, STORE is called for the ADVELM record.

SYNCLZ: First STRMAK is called to obtain the verb and DCTFND is called to validate. Similarly, STRMAK and DCTFND are called for the adverb. Next SEEKER is used to assure that no duplicate SYNCLZ record exists and then is used to find the adverb on the ADVADV chain. Finally, STORE is called.

MODULE

Each card image calls for an entry to the module link table. This table is retrieved at the first call to this section but is not modified until BOOT is terminating (see figure 21, statement 64). For each card image STRMAK is called to obtain the verb name and DCTFND is used to get the verb number. The link name is then stored in LINKS indexed by the verbs number. A valid card for this section is always marked as a change.

HEADER

Each card image calls for a new header to be created. First, RNMFND is called to validate the record type name. Next, HDFND is called to make sure no duplicate exists. If the record type name is TGTHD, NUMFND is called to set ICLASS. Finally, HDPUT is called to create the header.

| Subroutine BOOT is illustrated in figure 21.

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

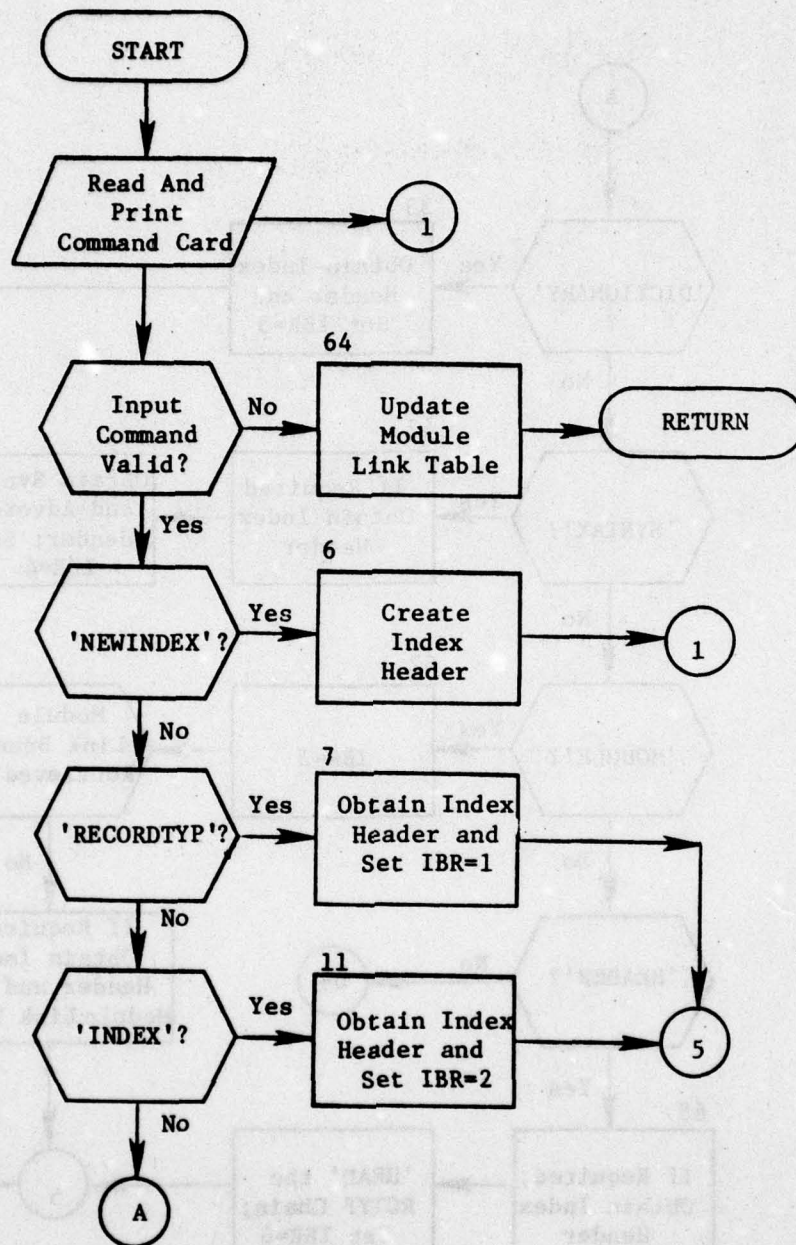


Figure 21. Subroutine BOOT (Part 1 of 24)

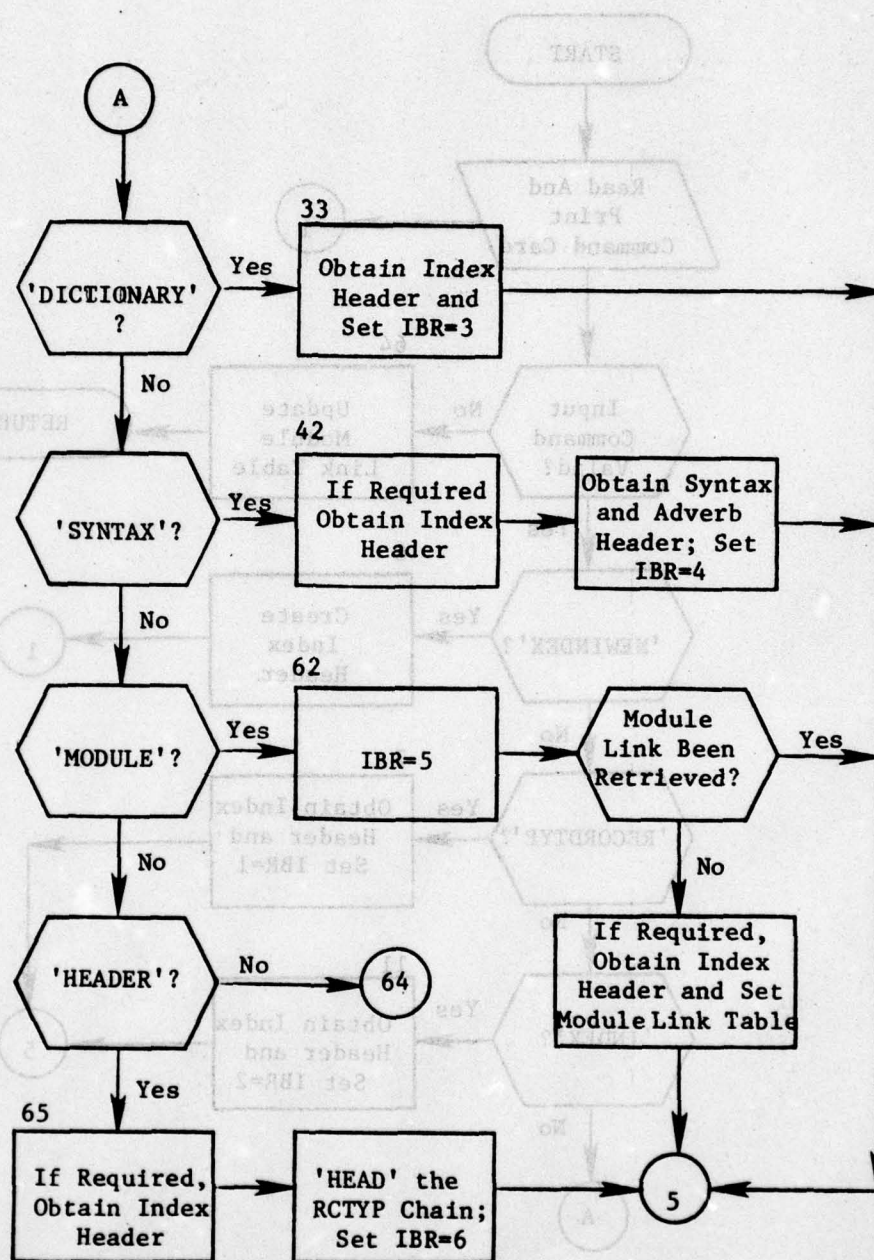


Figure 21. (Part 2 of 24)

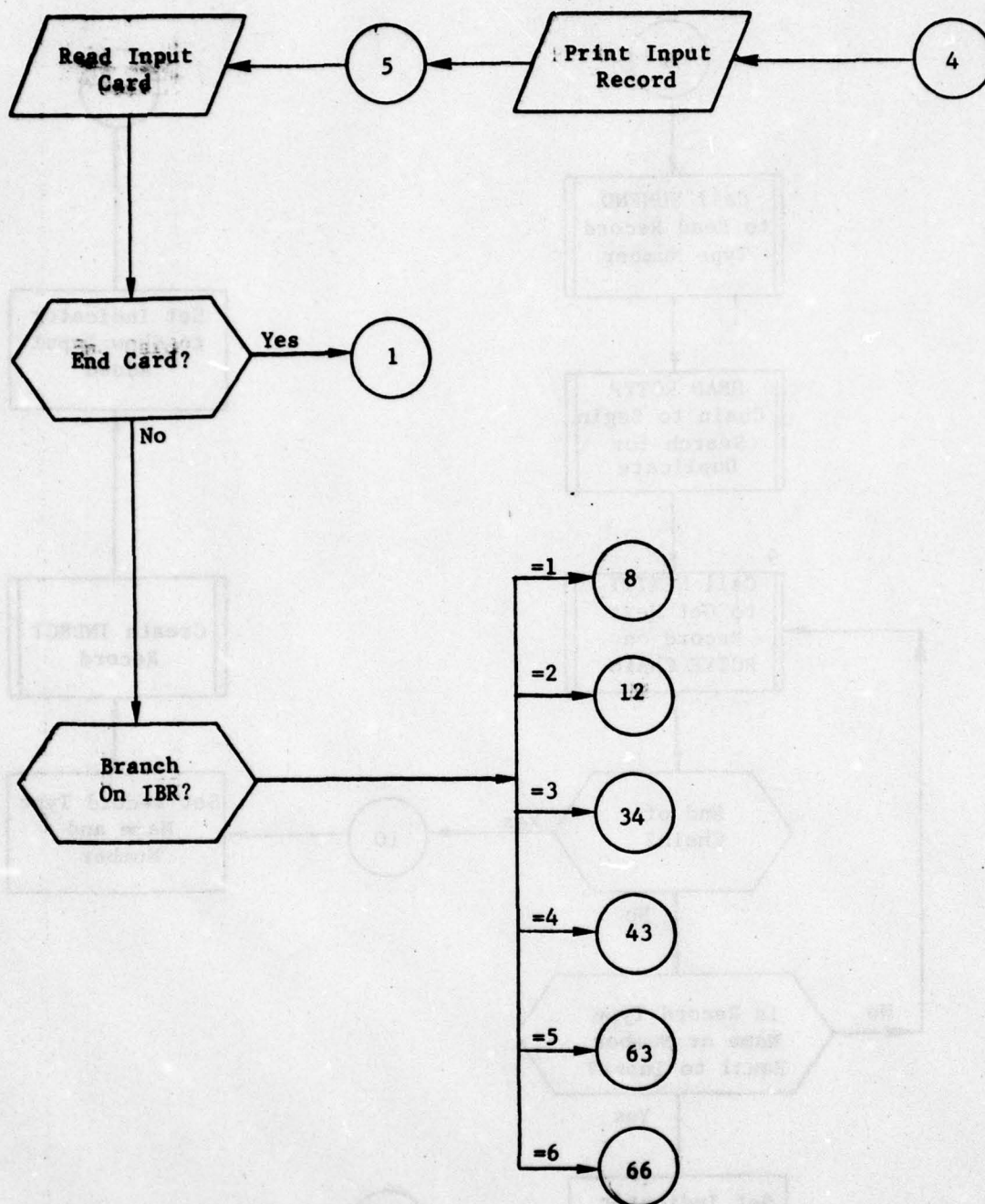


Figure 21. (Part 3 of 24)

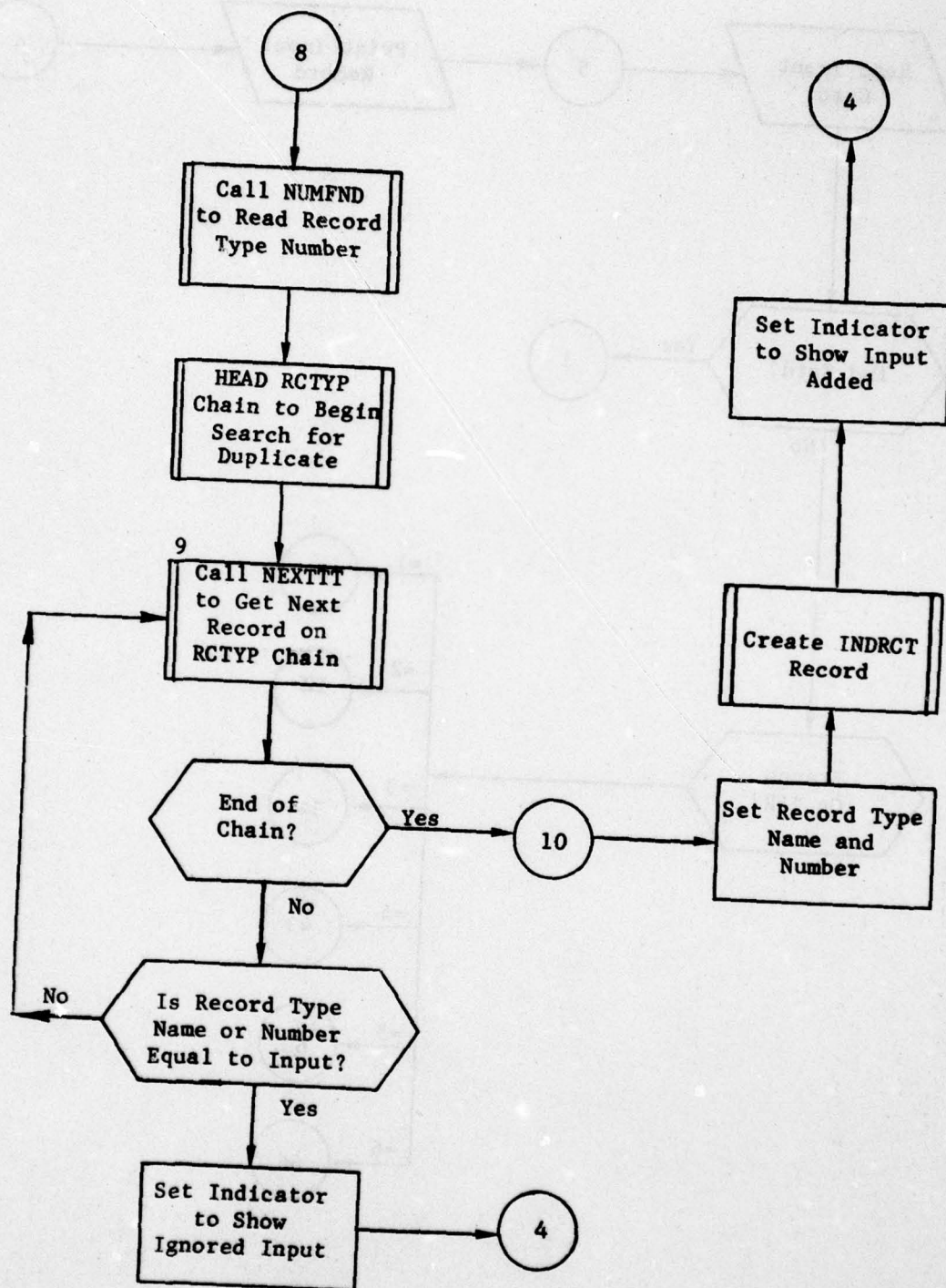


Figure 21. (Part 4 of 24)

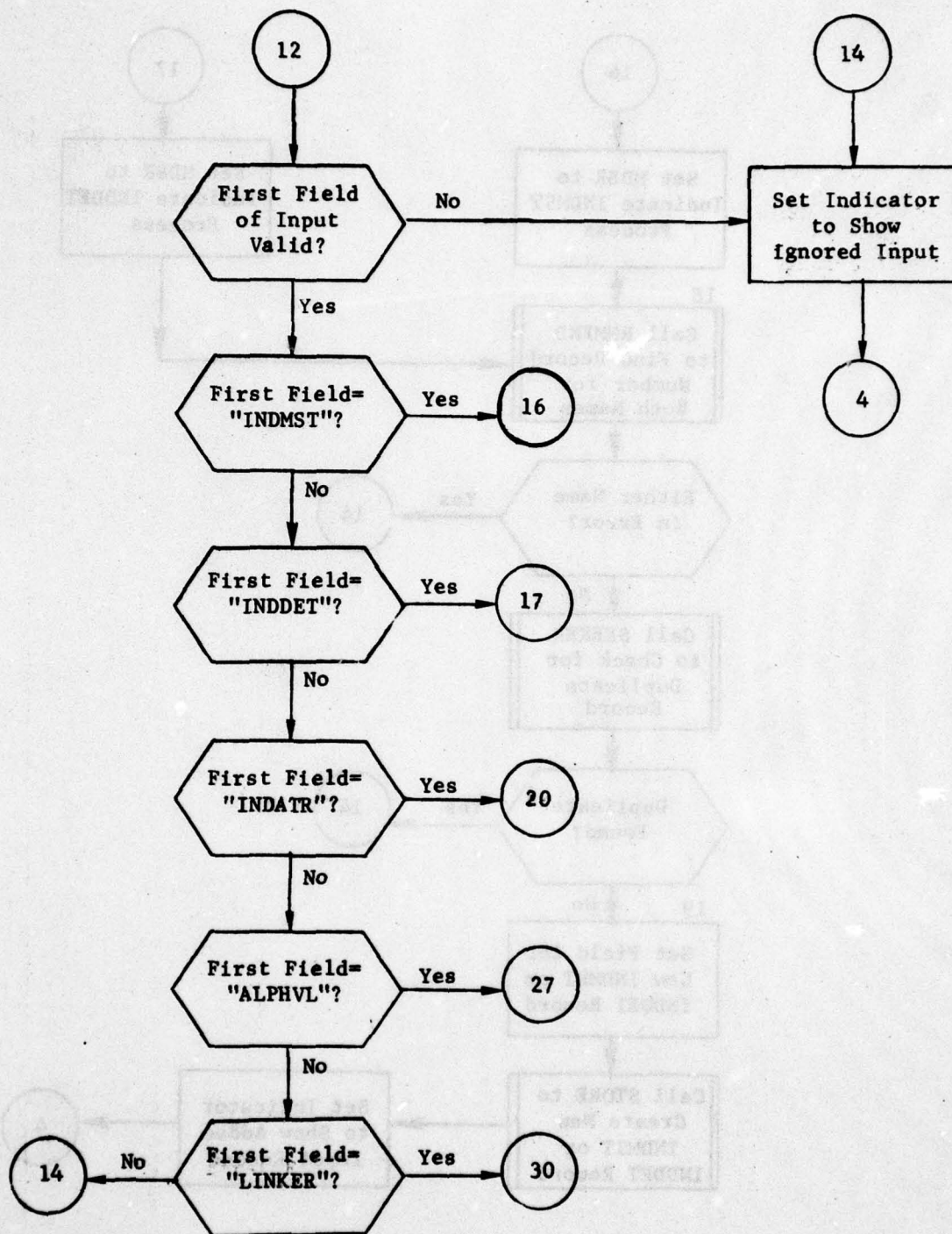


Figure 21. (Part 5 of 24)

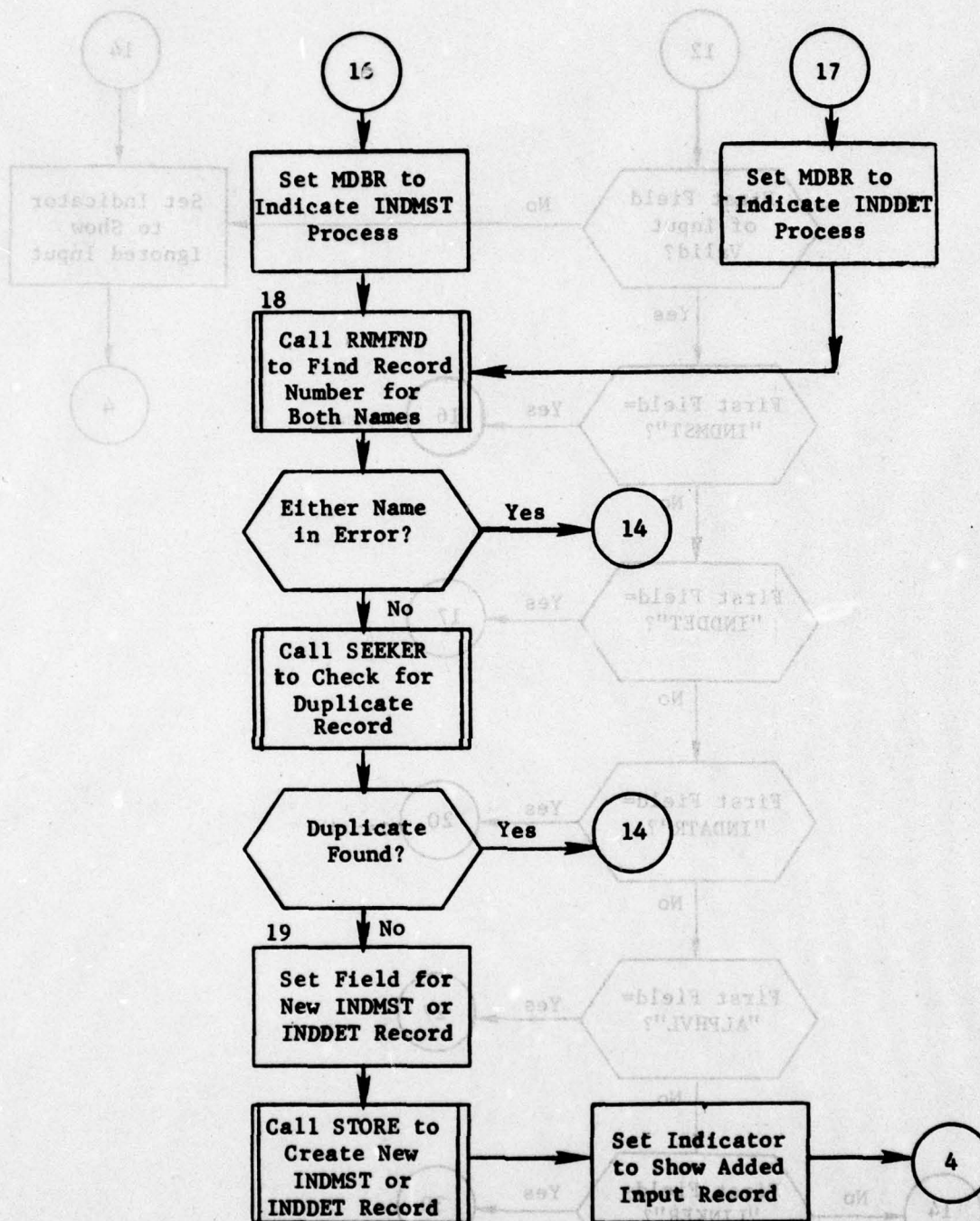


Figure 21. (Part 6 of 24)

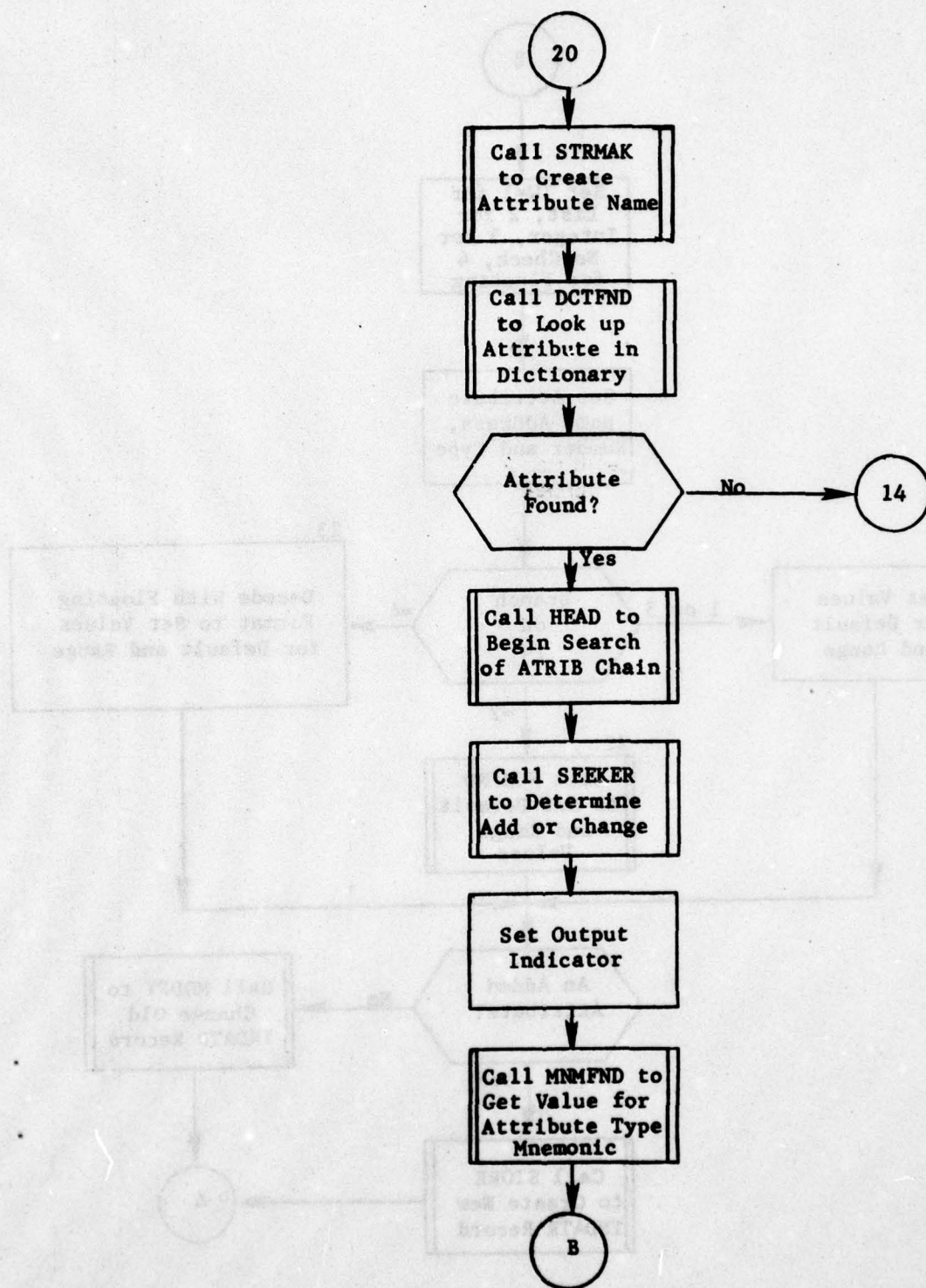


Figure 21. (Part 7 of 24)

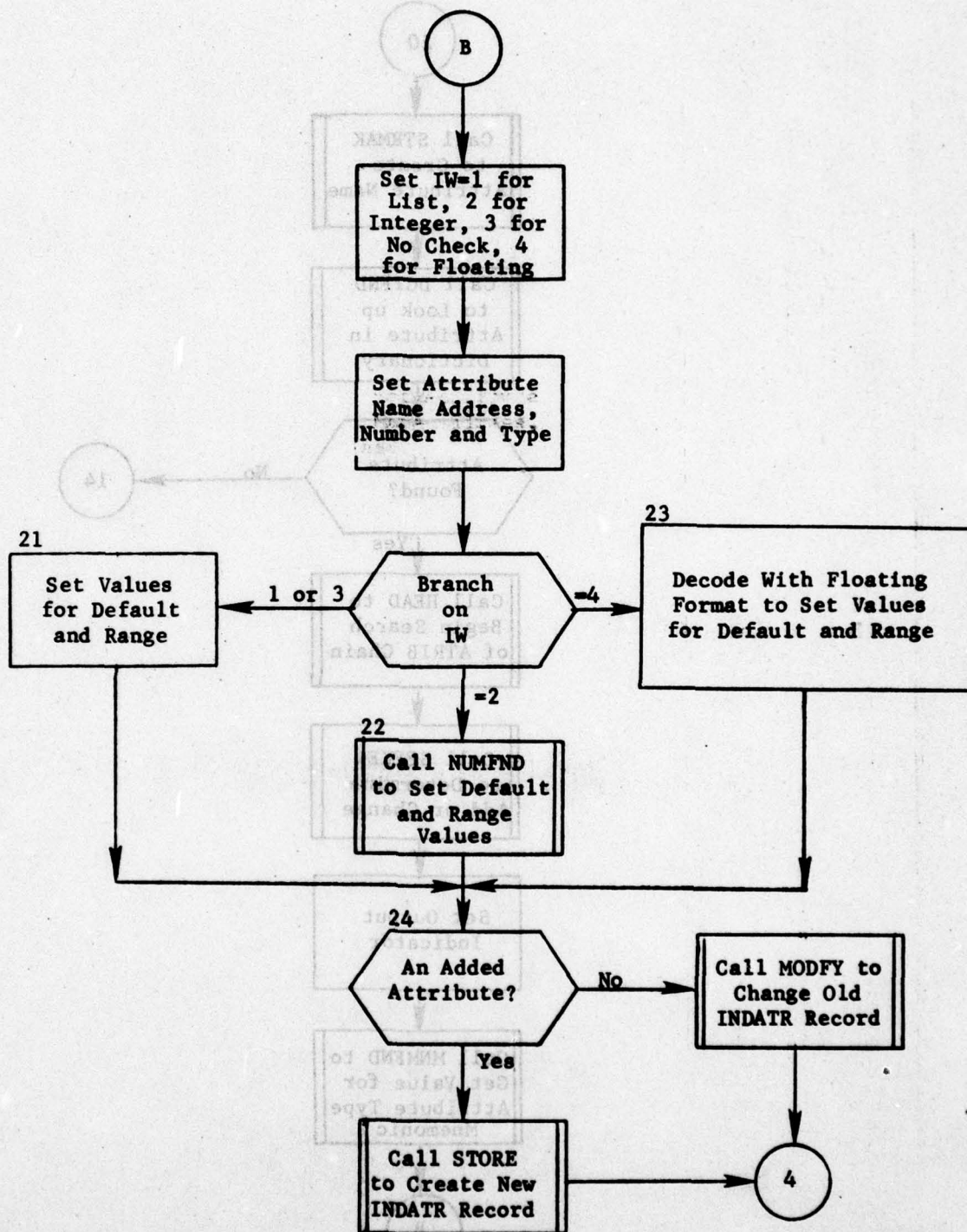


Figure 21. (Part 8 of 24)

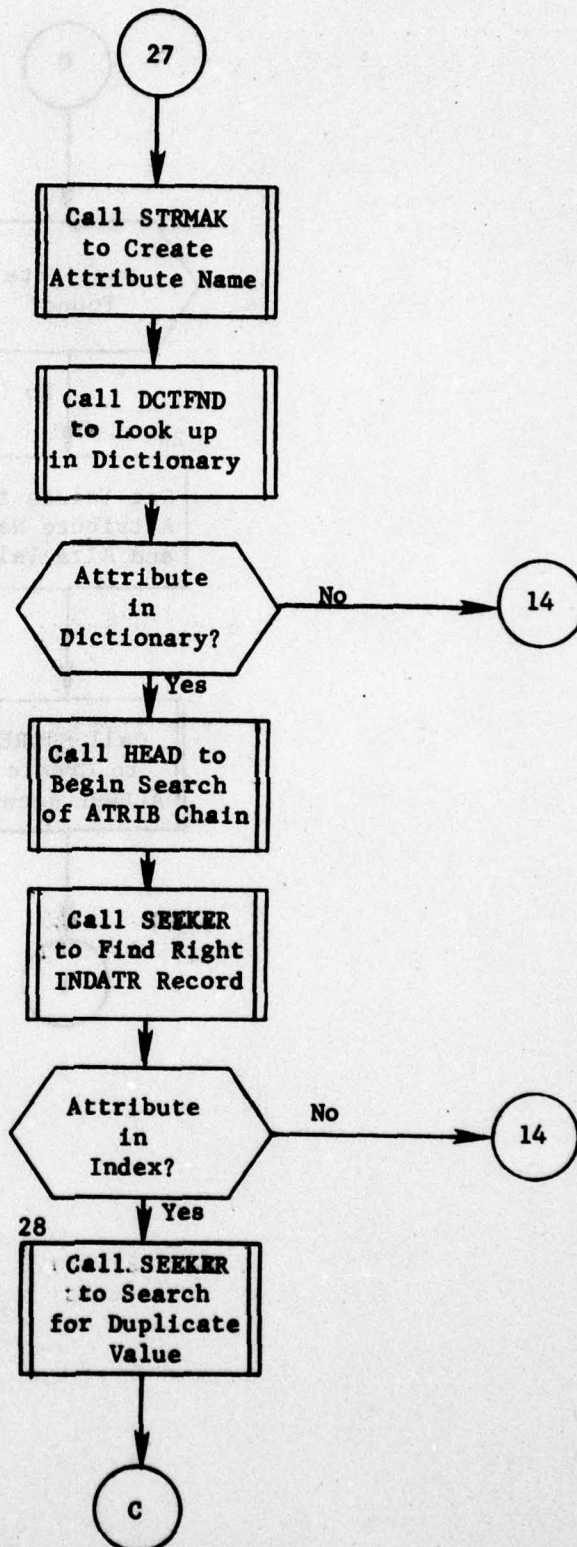


Figure 21. (Part 9 of 24)

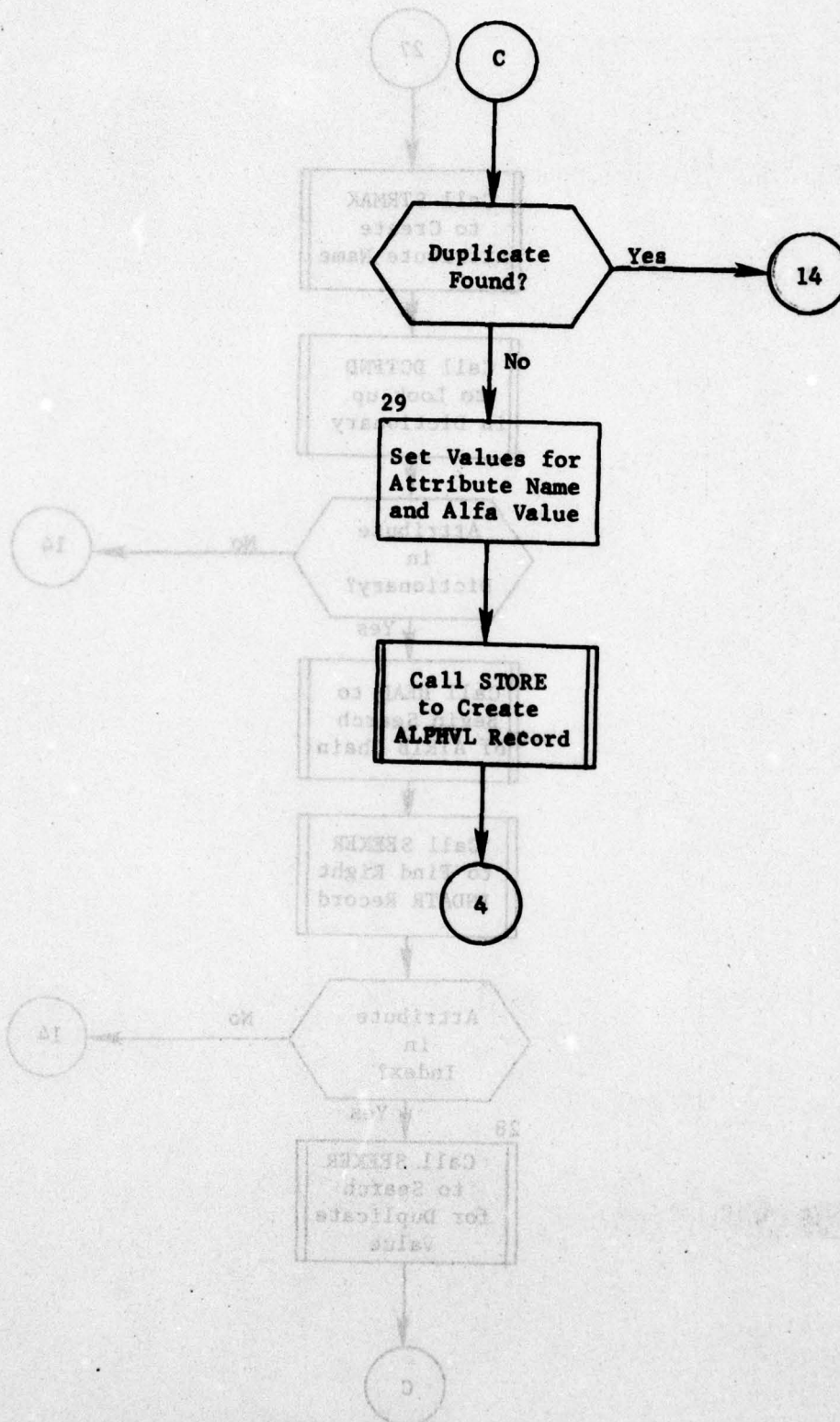


Figure 21. (Part 10 of 24)

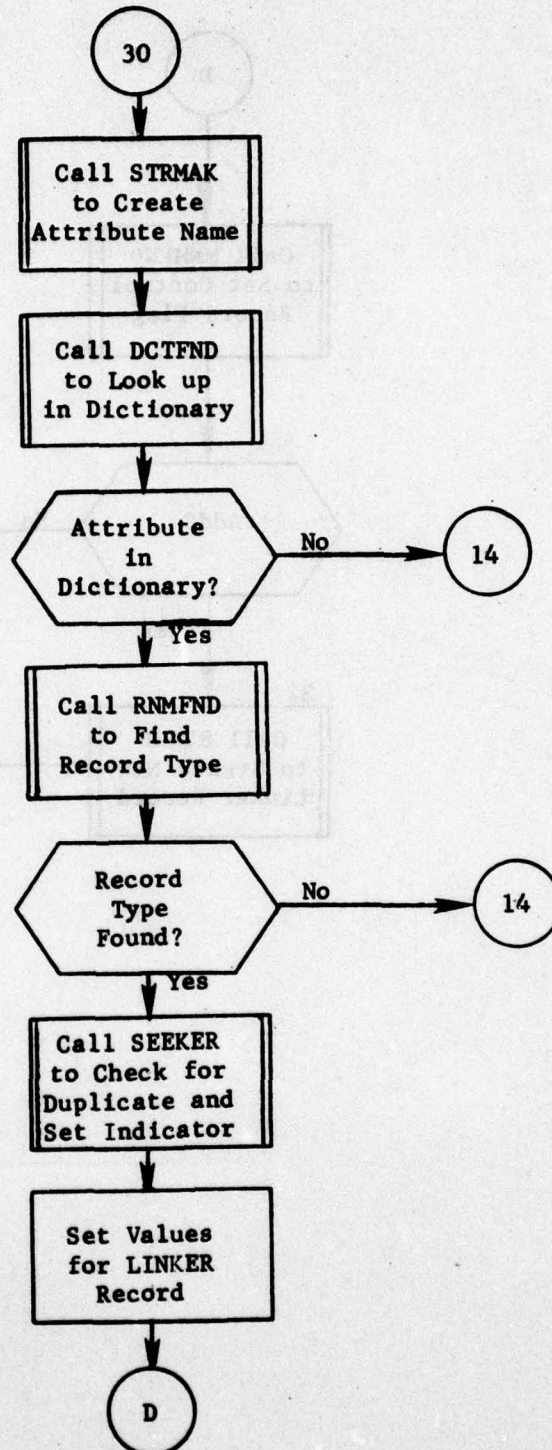


Figure 21. (Part 11 of 24)

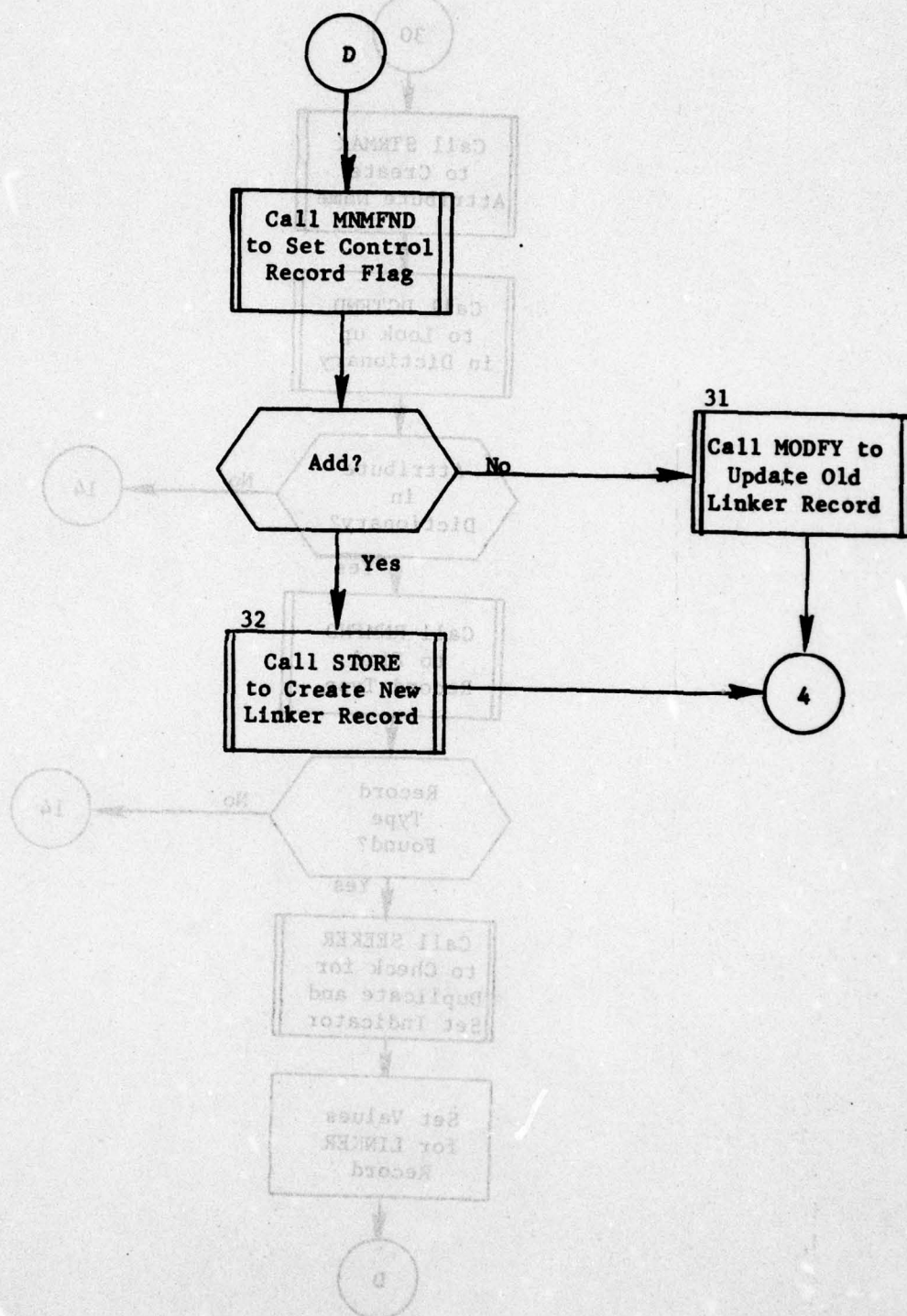


Figure 21. (Part 12 of 24)

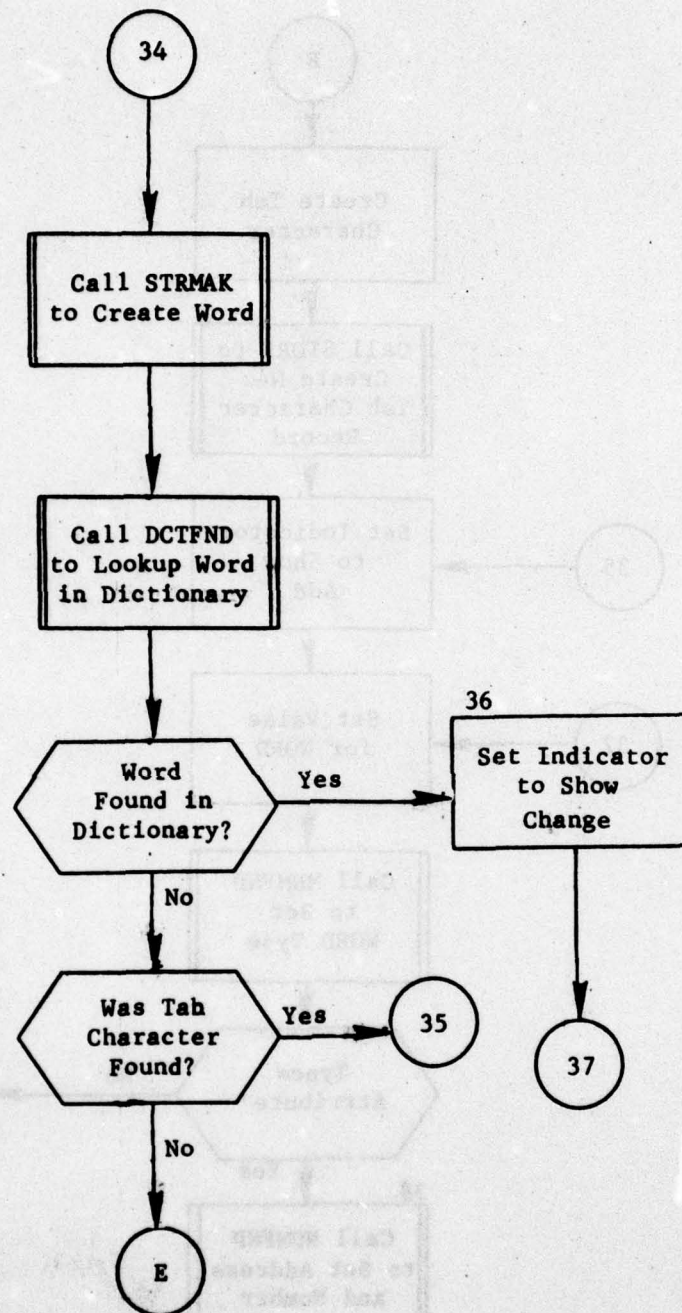


Figure 21. (Part 13 of 24)

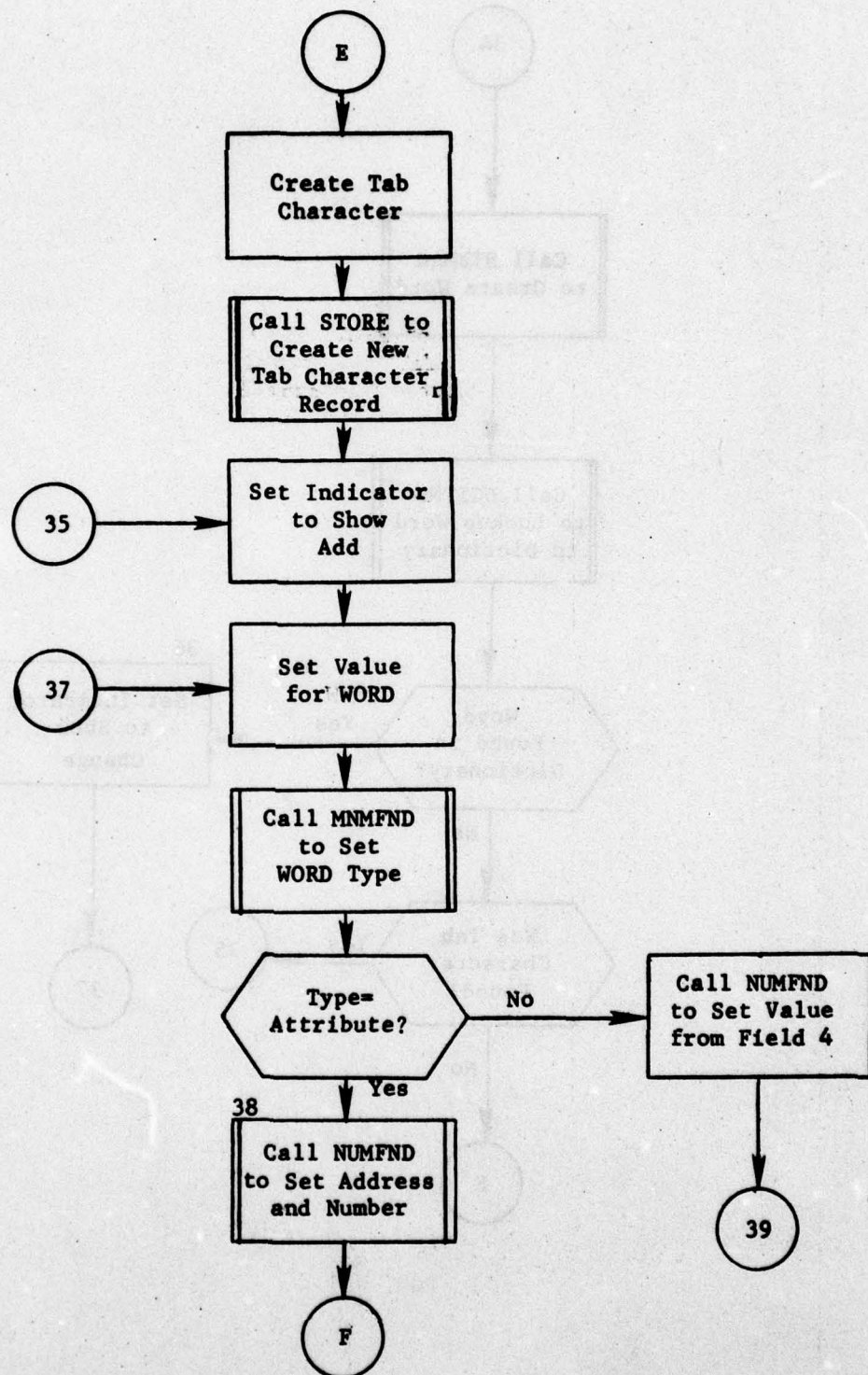


Figure 21. (Part 14 of 24)

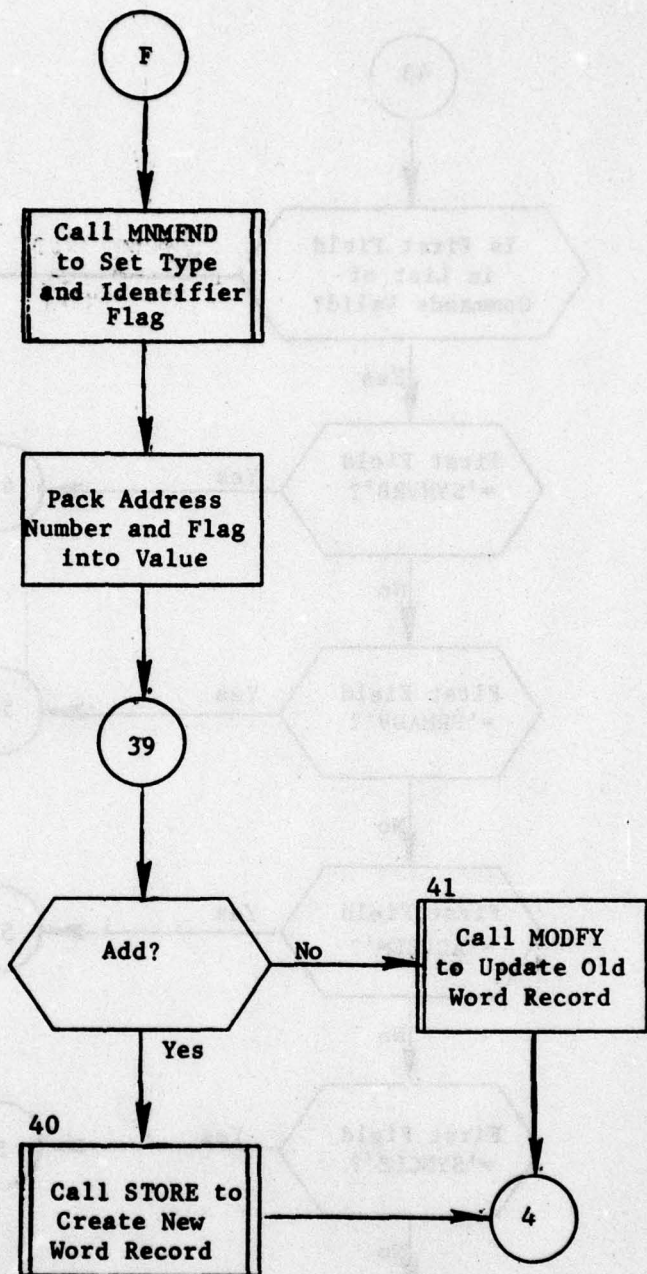


Figure 21. (Part 15 of 24))

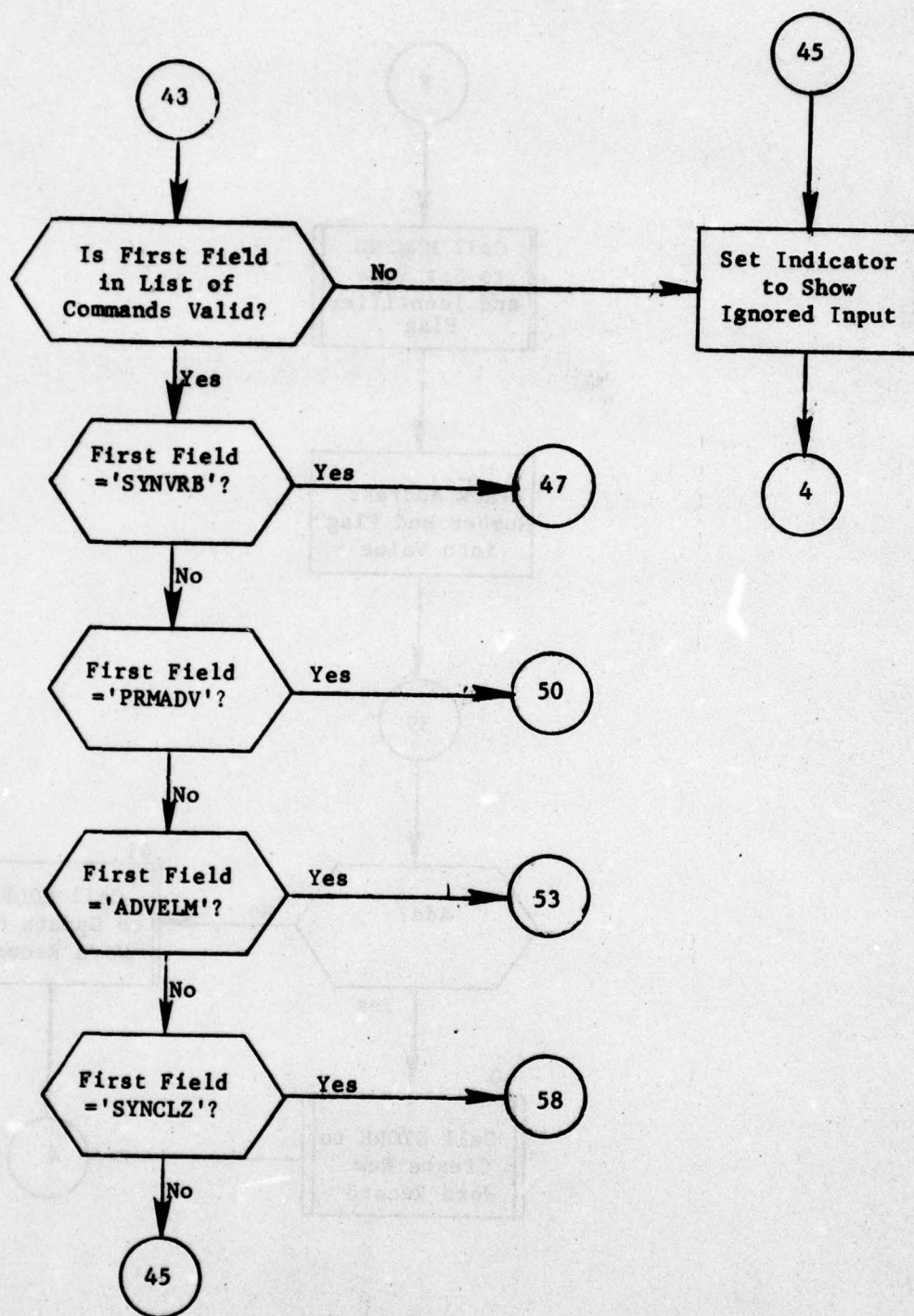


Figure 21. (Part 16 of 24)

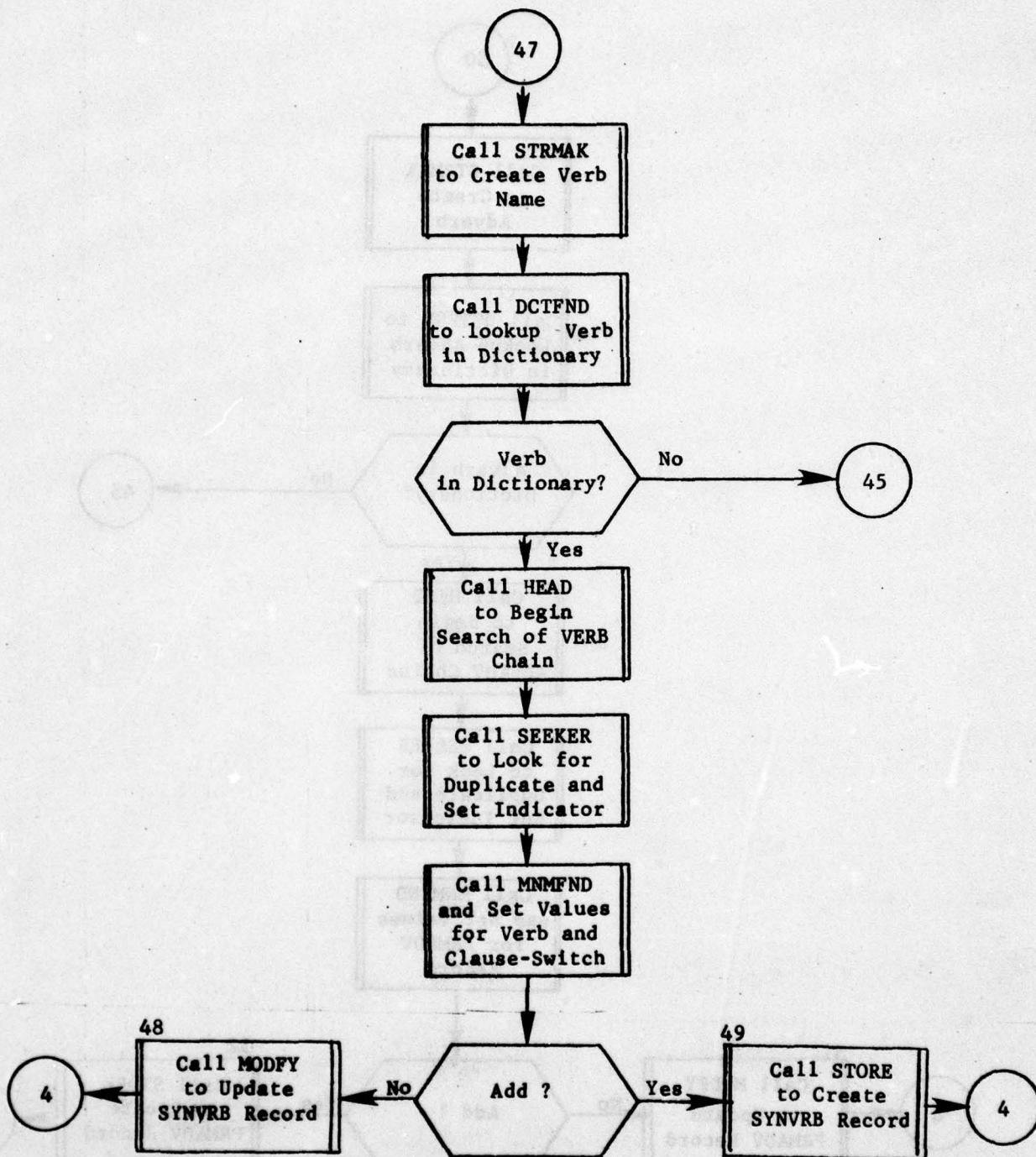


Figure 21. (Part 17 of 24)

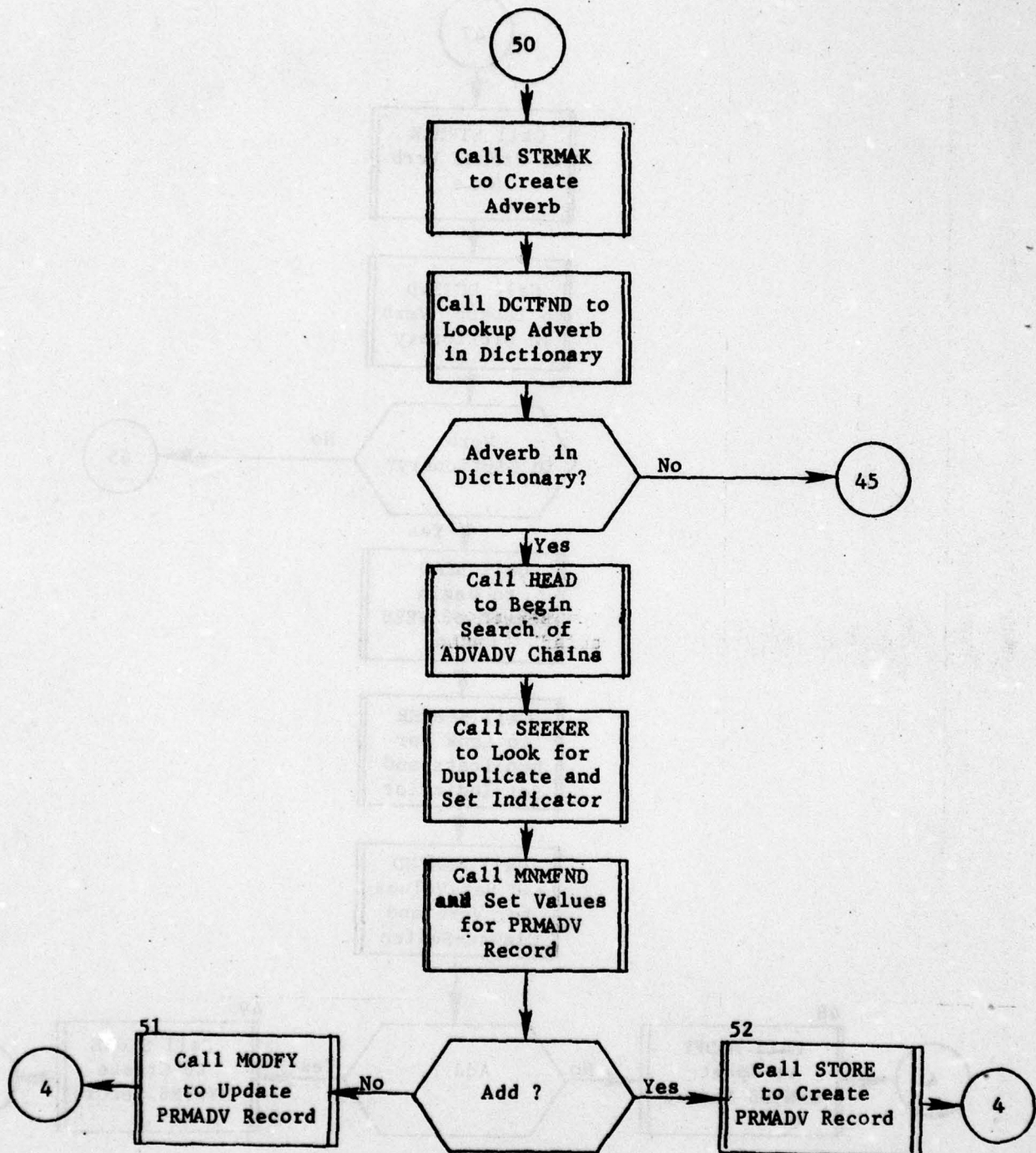


Figure 21. (Part 18 of 24)

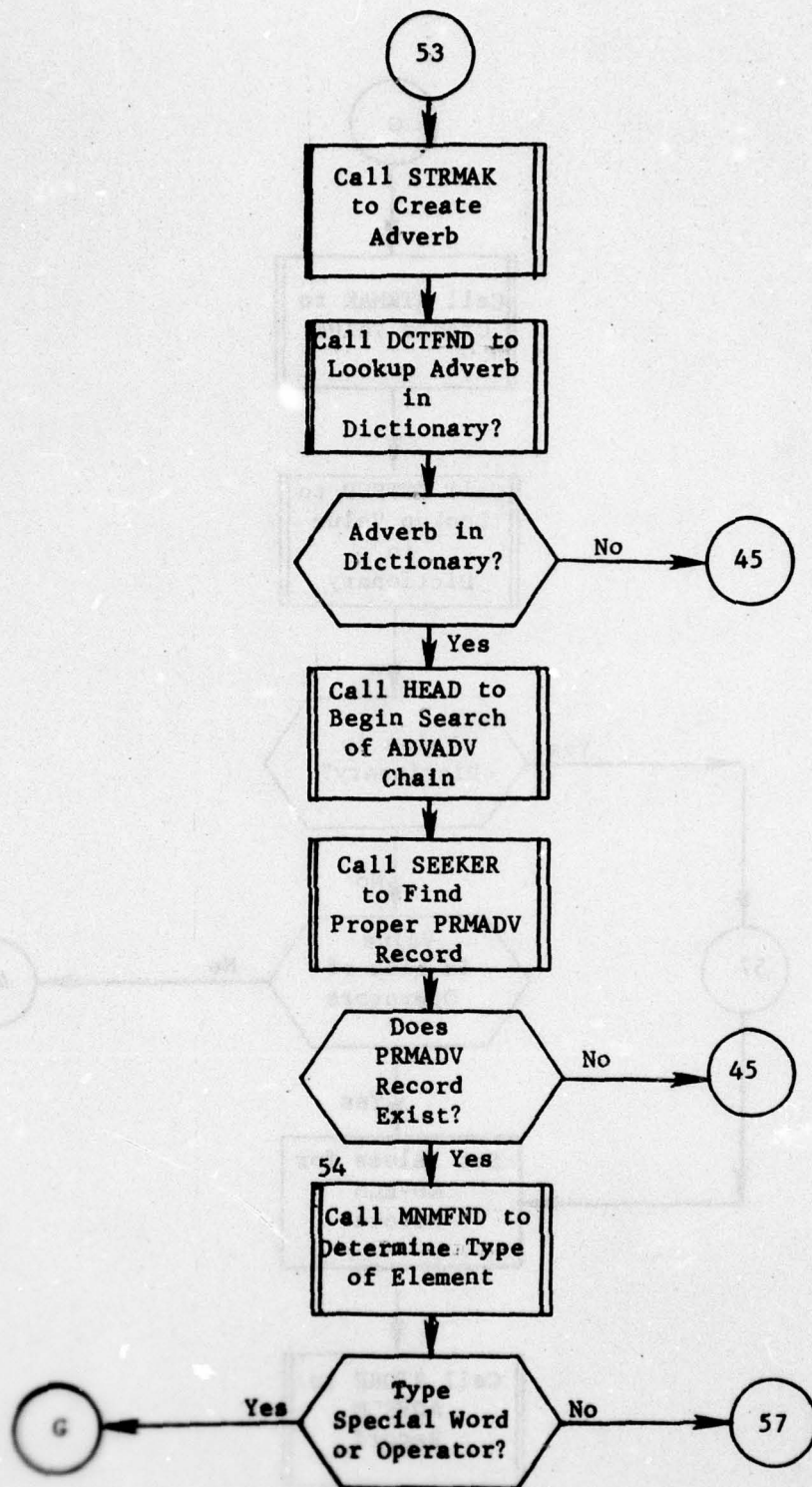


Figure 21. (Part 19 of 24)

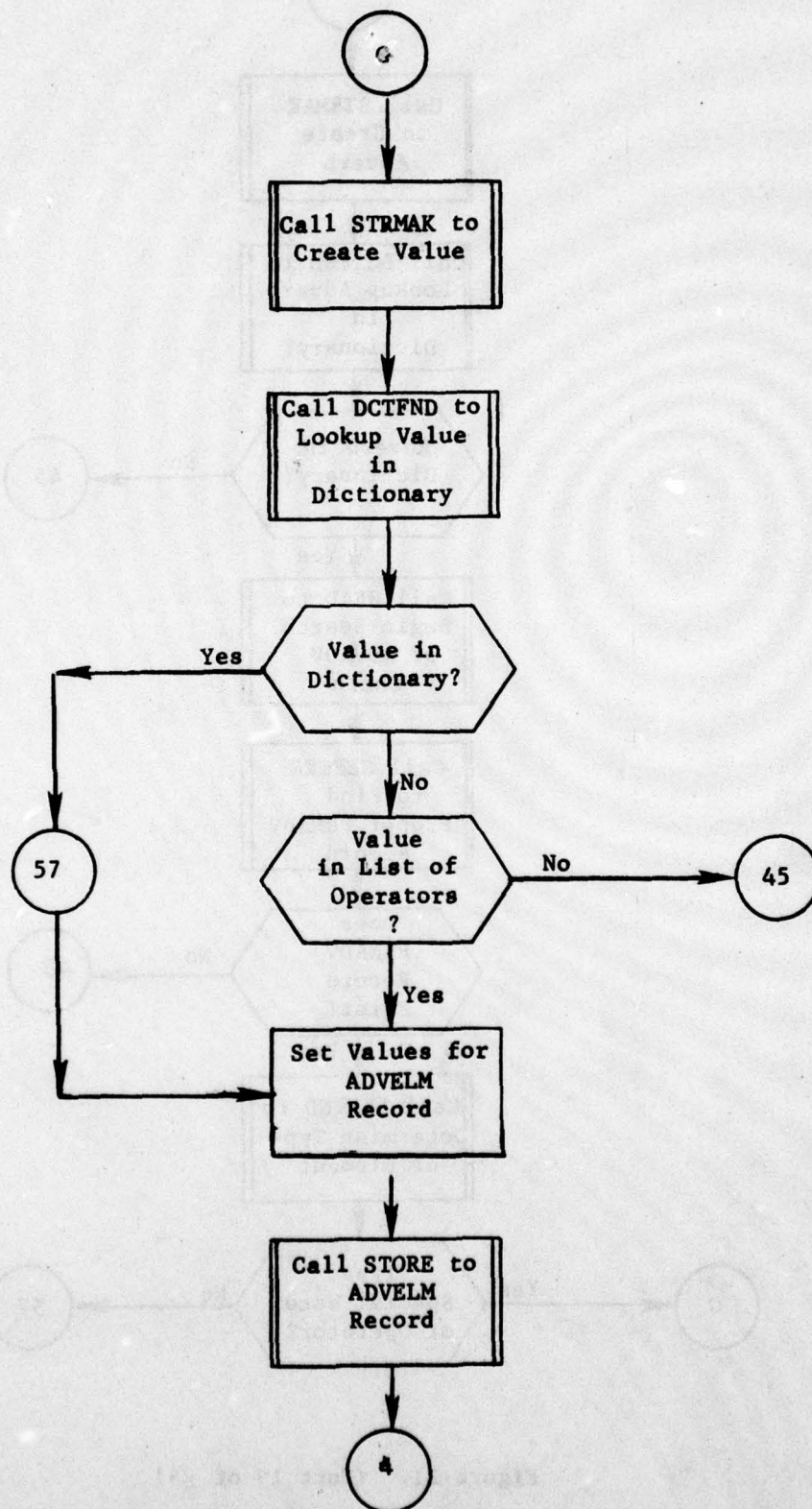


Figure 21. (Part 20 of 24)

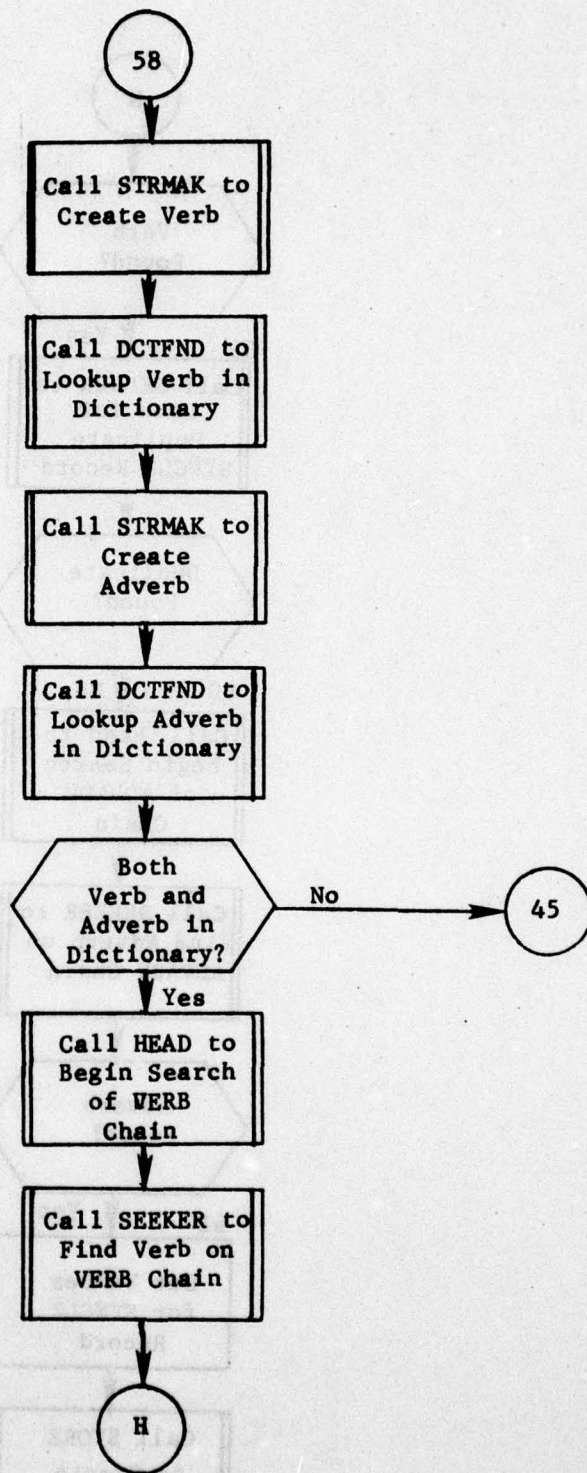


Figure 21. (Part 21 of 24)

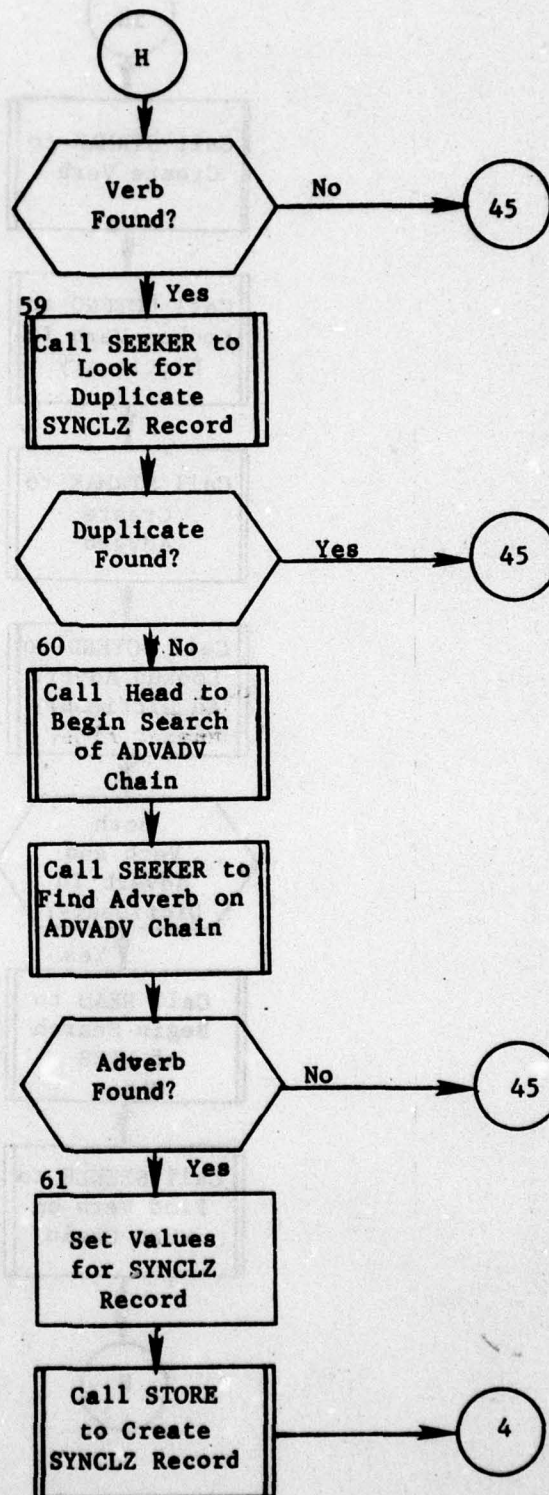


Figure 21. (Part 22 of 24)

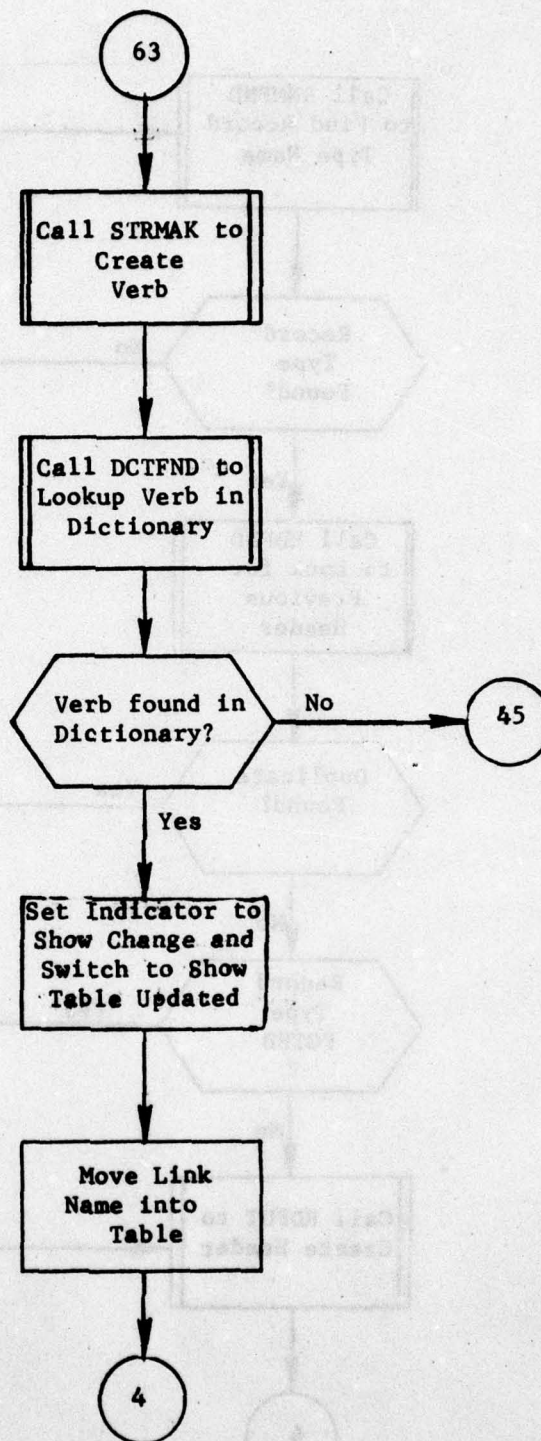


Figure 21. (Part 23 of 24)

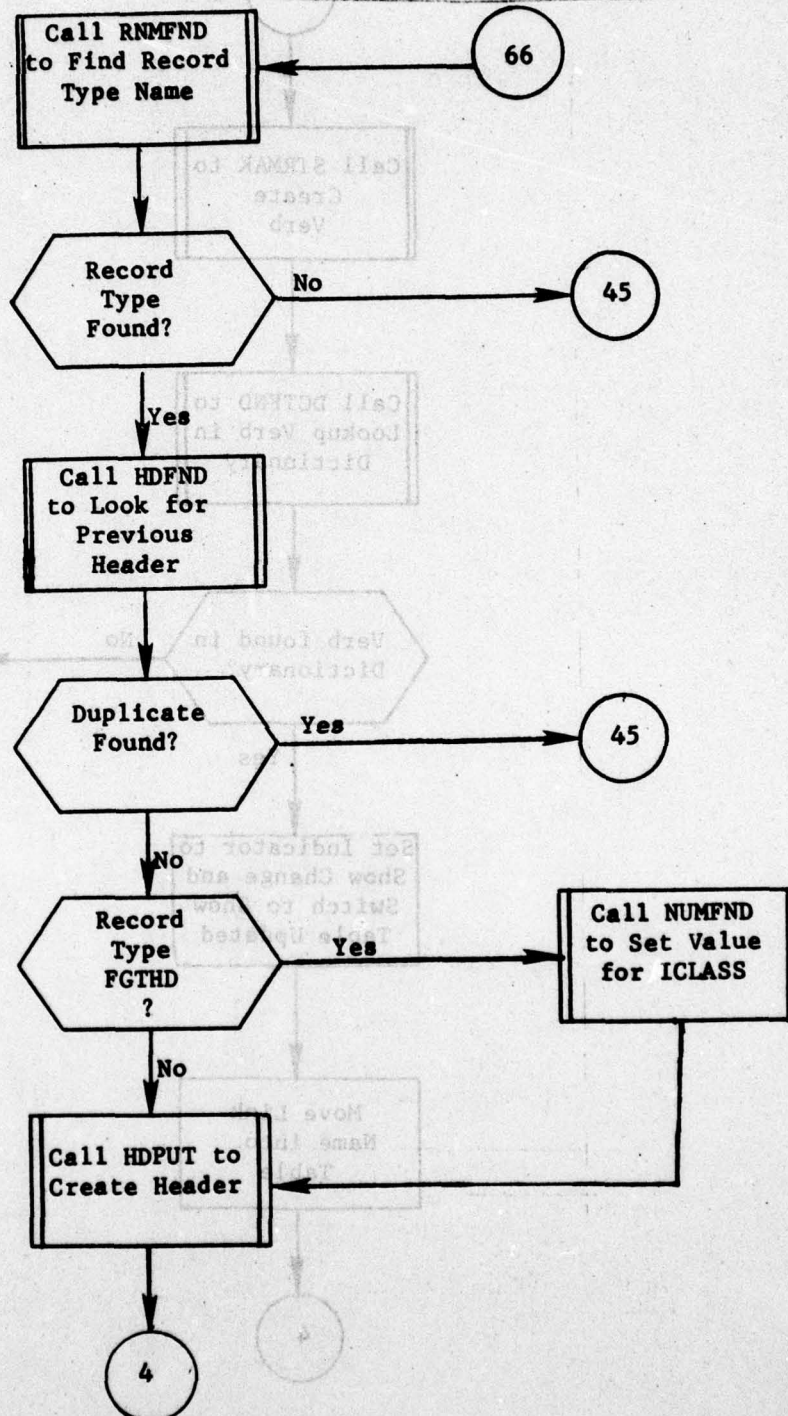


Figure 21. (Part 24 of 24)

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

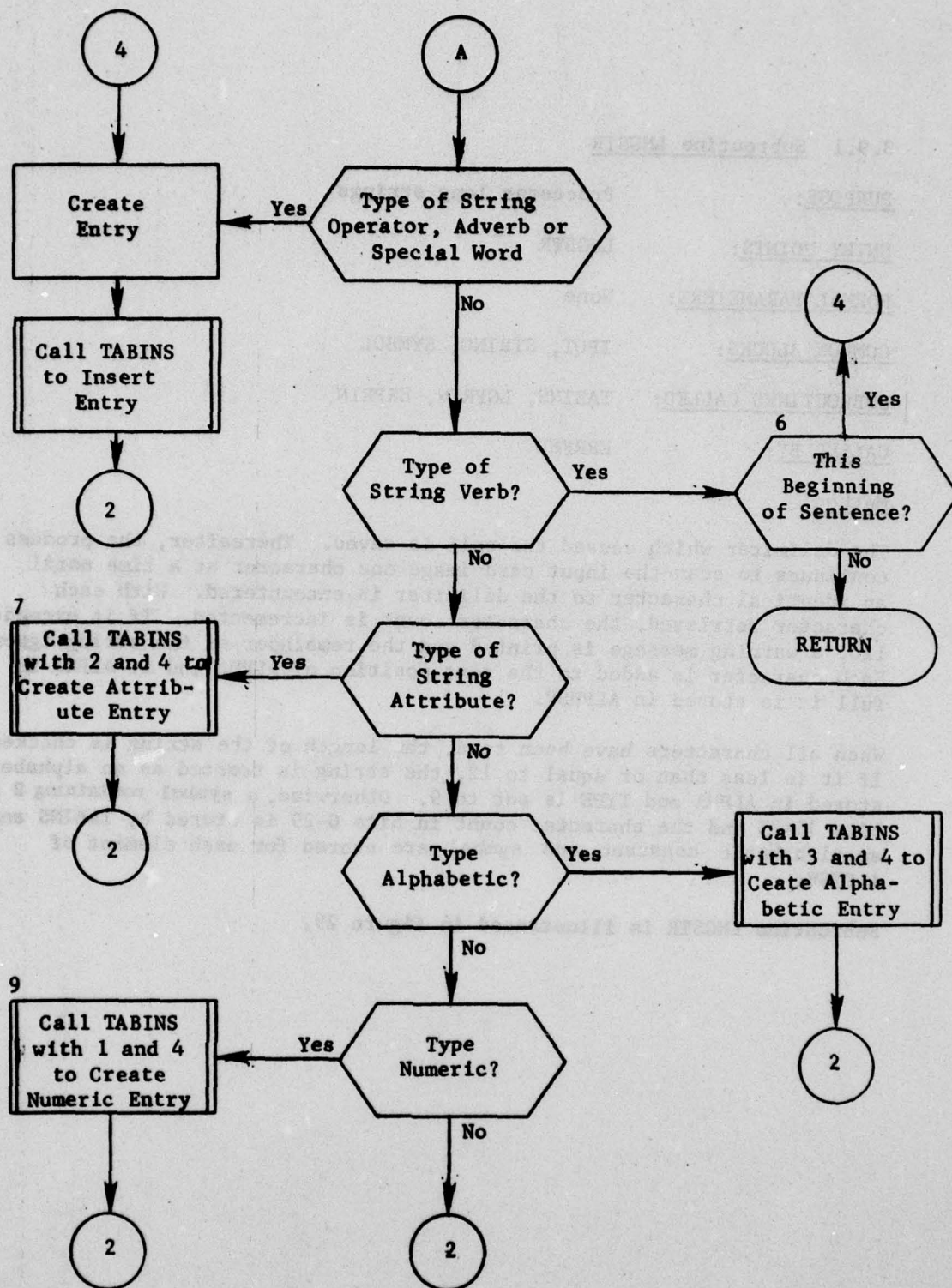


Figure 28. (Part 2 of 2)

3.9.1 Subroutine LNGSTR

PURPOSE: Processes long strings

ENTRY POINTS: LNGSTR

FORMAL PARAMETERS: None

COMMON BLOCKS: IPQT, STRING, SYMBOL

SUBROUTINES CALLED: TABINS, LGPRIN, ERPRIN

CALLED BY: ERRFND

Method:

The delimiter which caused the call is saved. Thereafter, the process continues to scan the input card image one character at a time until an identical character to the delimiter is encountered. With each character retrieved, the character count is incremented. If it exceeds 120, a warning message is printed and the remainder of the string ignored. Each character is added to the next position of ALPHA and if ALPHA is full it is stored in ALPHSV.

When all characters have been read, the length of the string is checked. If it is less than or equal to 12, the string is treated as an alphabetic, stored in ALPHA and TYPE is set to 9. Otherwise, a symbol containing 2 in bits 30-35 and the character count in bits 0-29 is stored by TABINS and an alphabetic constant and symbol are stored for each element of ALPHSV.

Subroutine LNGSTR is illustrated in figure 29.

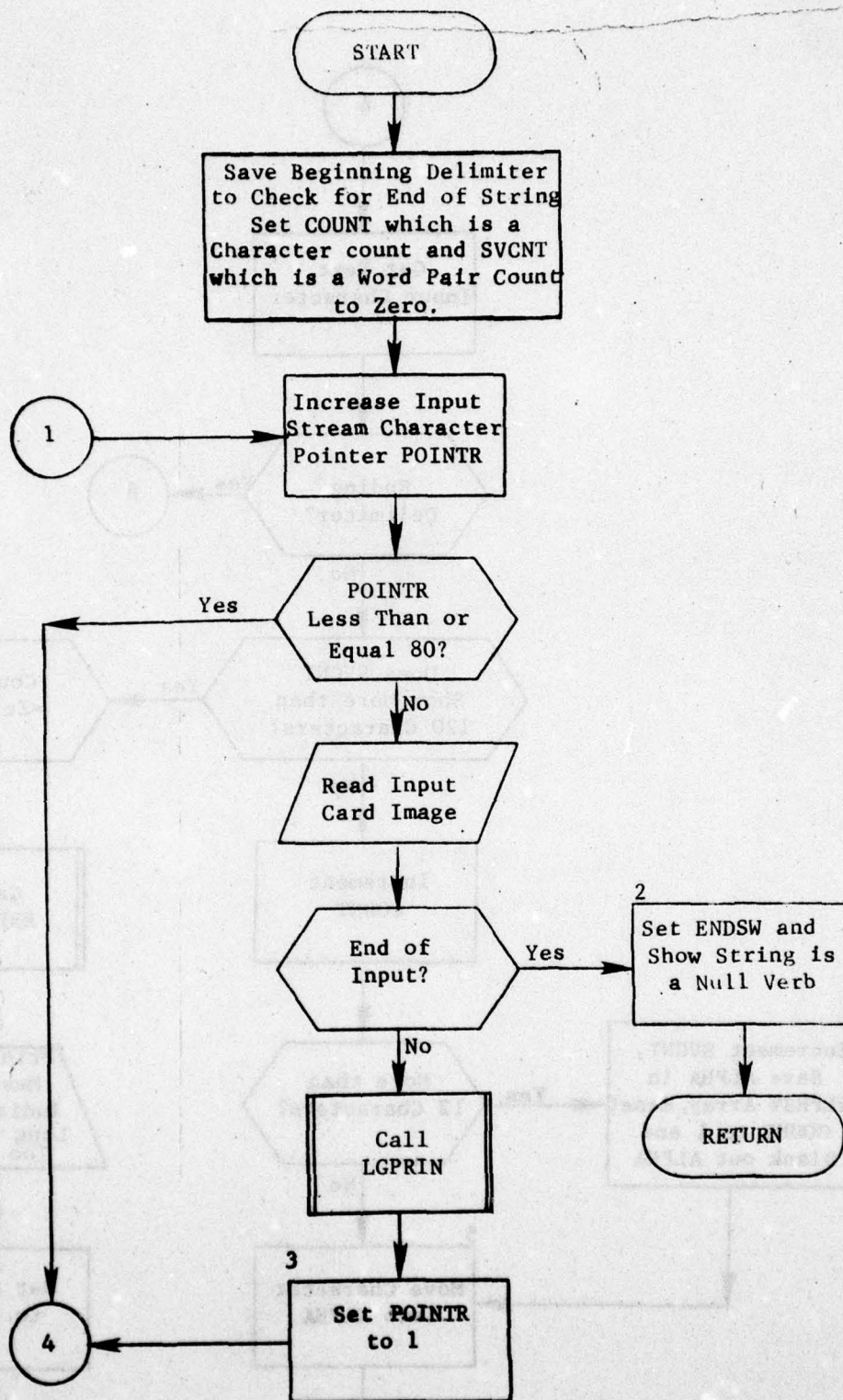


Figure 29. Subroutine LNGSTR (Part 1 of 4)

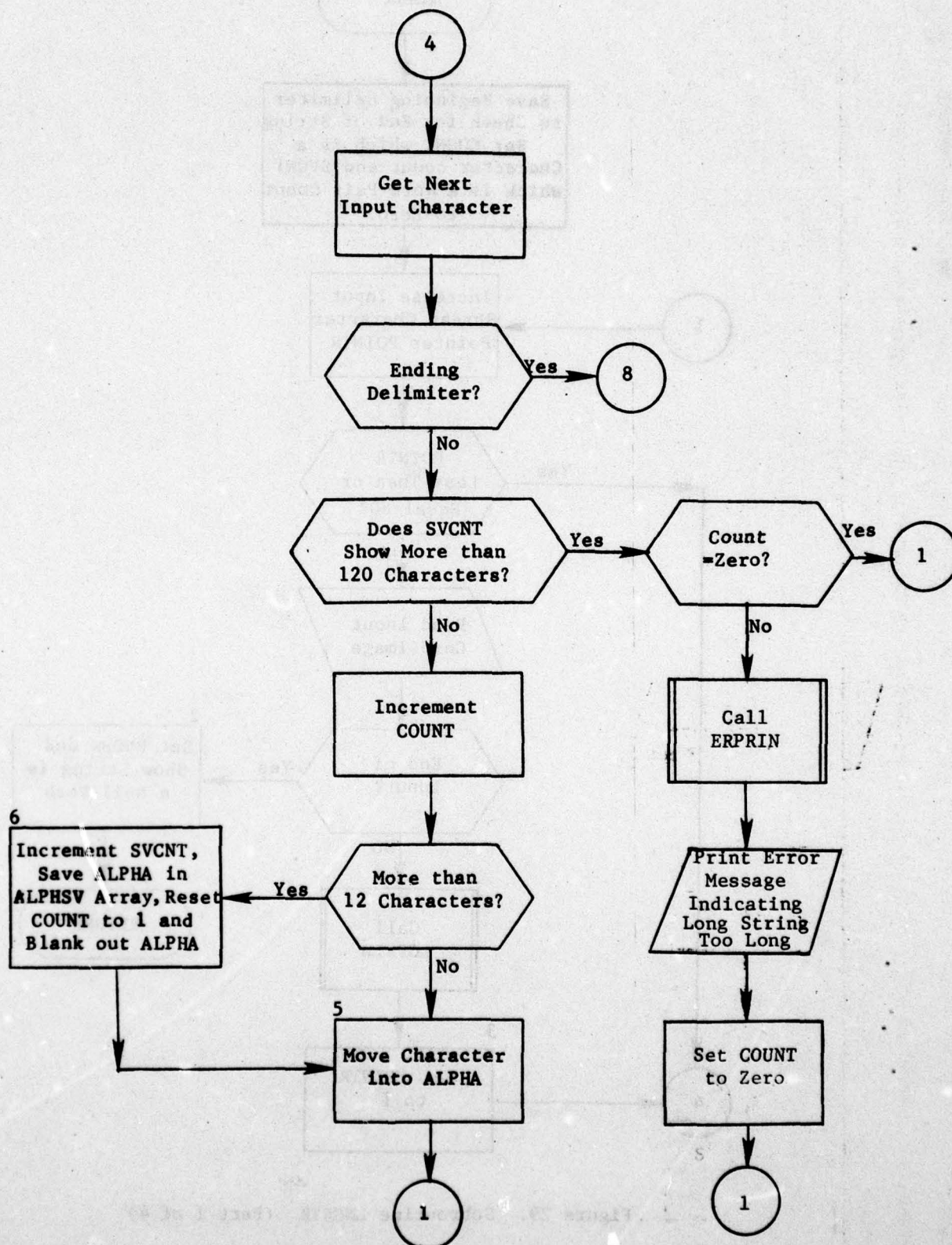


Figure 29. (Part 2 of 4)

3.9.2 Subroutine SYNTAX

PURPOSE: Analyze syntax of input command sentences

ENTRY POINTS: SYNTAX

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C30, FIRST, IPGT, OOPS, STRING

SUBROUTINES CALLED: ERPRIN, HDFND, HEAD, NEXTTT, RETRV

CALLED BY: ERRFND

Method:

This subroutine checks each input string against the current setting of various switches. If the string is one the types of strings expected, the appropriate switches are reset. The following switches and counters are used by SYNTAX:

- BETWEN - True when processing BETWEEN relation
- BLPRCT - Boolean parenthesis level count
- BOLTYT - True when expecting relational phrase, left parenthesis or NOT (boolean clause only)
- BOOL - True when clause is boolean
- CONTEQ - True when an equals relation is being continued in a boolean clause
- ELEMNT - True when phrases are elemental
- ELRT - 1 - if elemental non-boolean clause
2 - if elemental boolean clause
- ENDCLZ - True if clause may end
- ENDCOM - True if sentence may end
- EQUALS - True when processing EQUALS relation
- INCLZ - True when processing a clause
- INCOM - True when processing a sentence
- LIKE - True when processing a LIKE relation
- PHPNT - Relational phrase branch
 - 1 - expecting operation unless inside collection
 - 2 - expecting value or collection
 - 3 - check for equal and between continuation
- PHPRCT - Relational phrase parenthesis count
- RELPHR - True when processing a relational phrase
- RESTRC - True when phrase type is restricted relational (EQUALS or LIKE)
- SINGLE - True when clause type is single
- SNGCT - Count of phrases for single type clause

VALCT - Value element branch
 1 - expecting end of element or OF
 2 - expecting identifying attribute
 3 - expecting alphabetic or numeric constant
 VALEL - True if processing value element
 VALQE - True if processing value expression
 VALTYP - True when expecting new value element or left parenthesis
 VLPRCT - Value expression parenthesis count

Processing proceeds as follows:

First the string is checked to see if it is a verb, if not, branch to statement 5 (see figure 30). If so INCOM is checked. If INCOM is true but ENDCOM is false an error has occurred. Otherwise INCOM is set to false and the subroutine exists. If INCOM is false, the VERB chain is searched to find a match. When the match is found INCOM and ENDCOM are set to two and INCLZ is set to false. If the clause switch indicates that the verb must have a clause (ICSW=1), ENDCOM is set to false.

Statement 5 (figure 30)

If the string is not an adverb, a branch is made to statement 19. If it is an adverb INCOM is checked and if false an error has occurred. Next, if INCLZ is true but ENDCLZ is false an error has occurred. Next, the CLAUSE chain is searched to see if this adverb is matched to the current verb. If it is HEAD is called to relieve the adverb's record and the clause and phrase type (IXTYP and ILTYP) are packed into the adverb's value. Now various switches are set depending on the clause type (IXTYP). If boolean, set SINGE to false, BOOL to true, BLPRCT to zero and BOLTYP to true. If sequence, set SINGLE and BOOL to false. If single set SINGE to true, BOOL to false, SNGCT to zero. If null set INCLZ to false and ENDCLZ and ENDCOM to true. For all but the last, set INCLZ to true and ENDCLZ, ENDCOM and VALQE to false and branch on phrase type (ILTYP). For relational phrases set RESTRC to false, for restricted phrases set RESTRC to true, for both set RELPHR, ELEMNT, EQUALS, LIKE, BETWEN and CONTEQ to false, and PHPRCT to zero. For elemental phrases set ELEMNT to true.

Statement 19 (figure 30)

At this point refer to the flow chart, figure 30 as processing is best illustrated therein. However, some sections that are of particular note follow.

Statement 52 (figure 30)

Here many relational phrases have ended. The LIKE phrase has one more element--the value for the identifiers attribute. The BETWEEN phrase may have an optional 'AND'. After this or without it the BETWEN switch

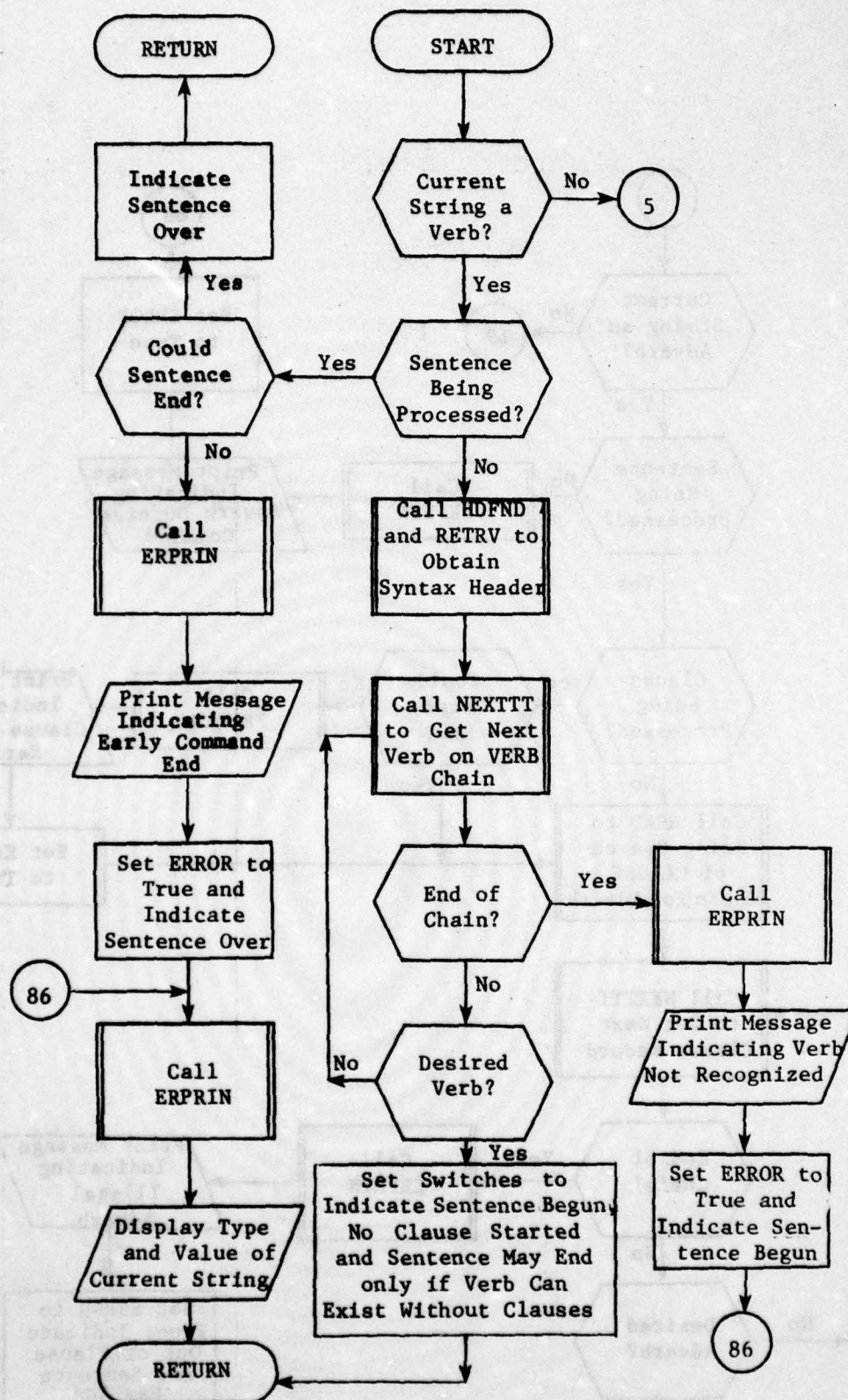
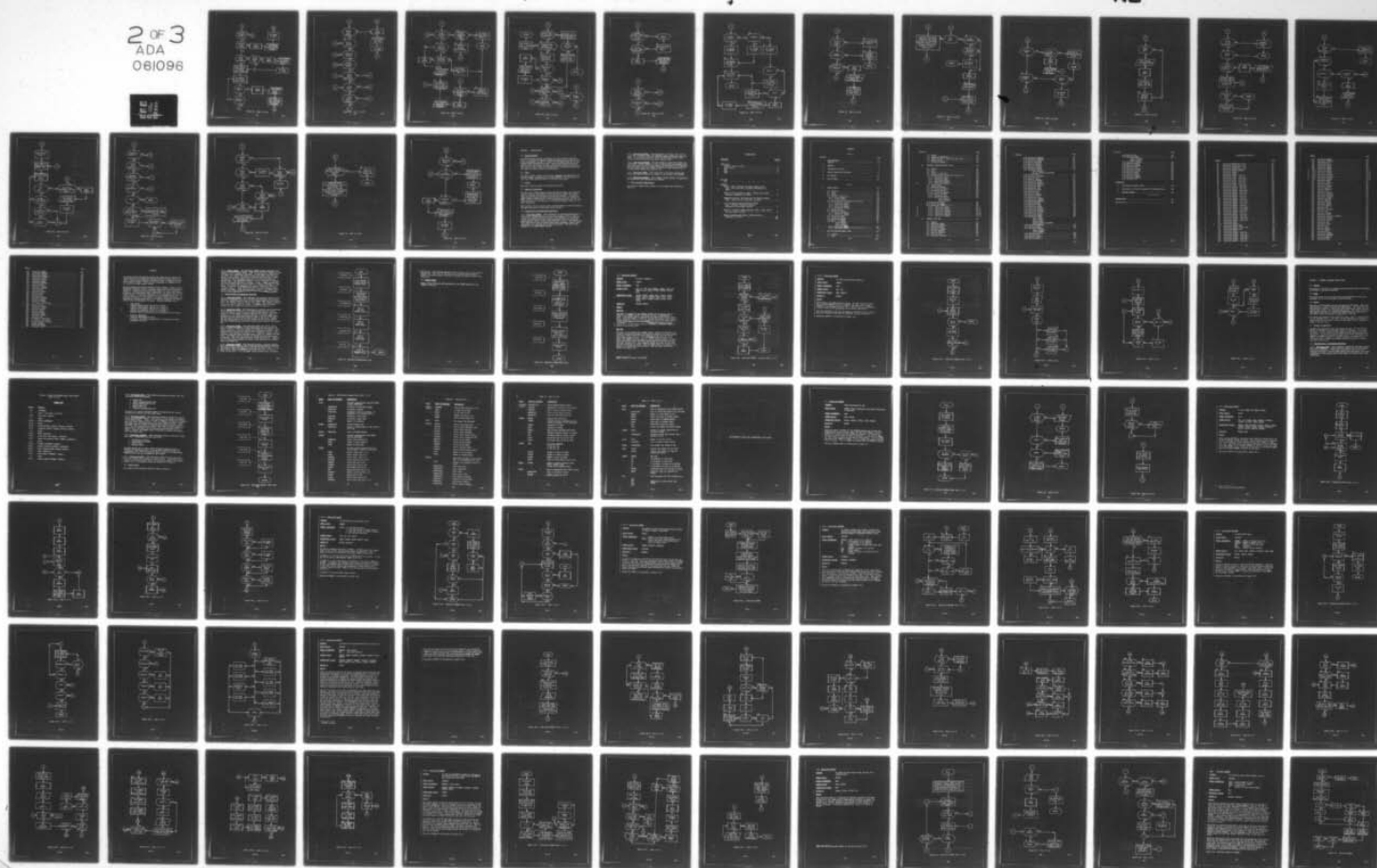


Figure 30. Subroutine SYNTAX (Part 1 of 22)

F/G 5/2
PRO--ETC(U)

NL

2 OF 3
ADA
061096



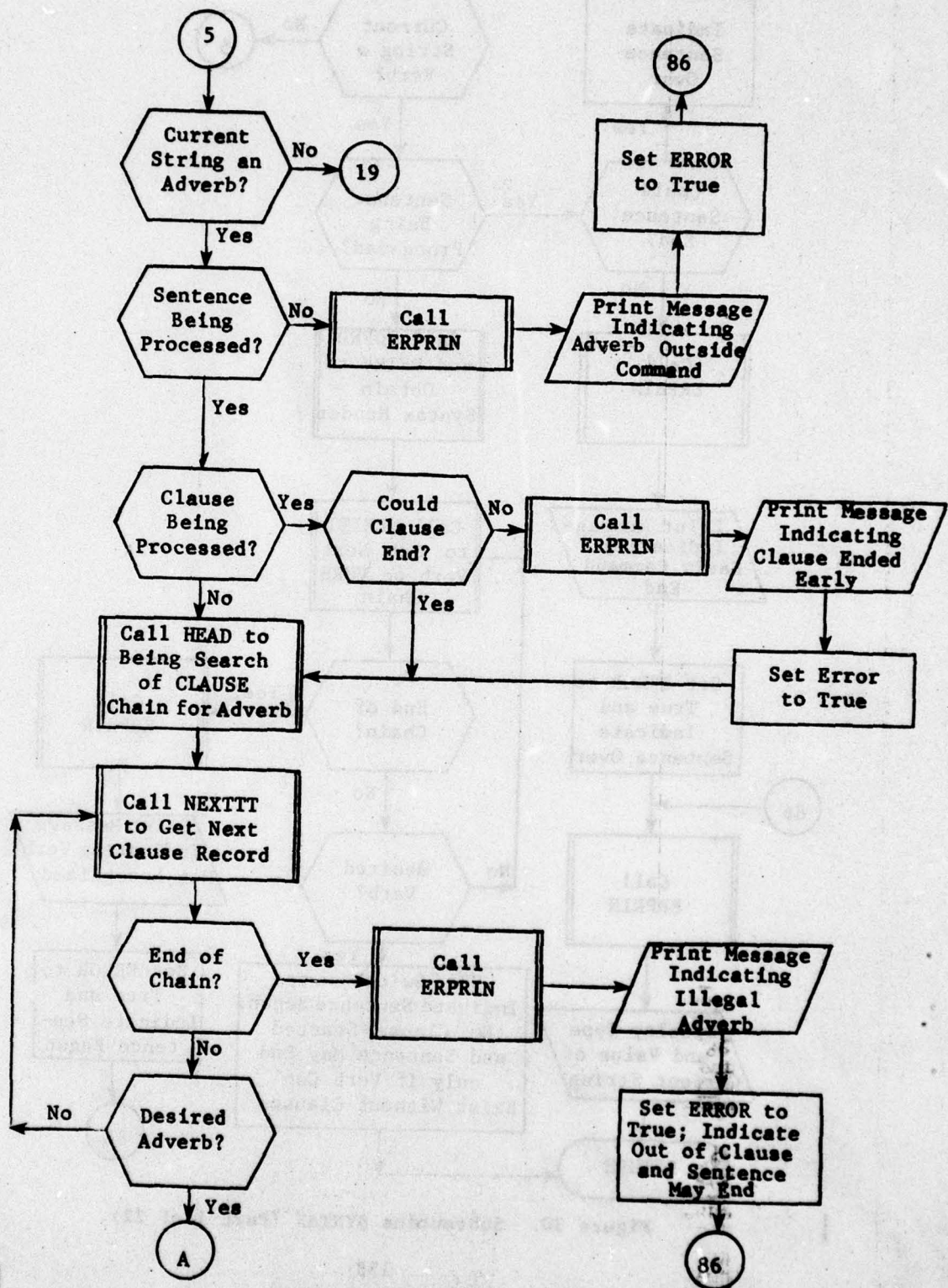


Figure 30. (Part 2 of 22)

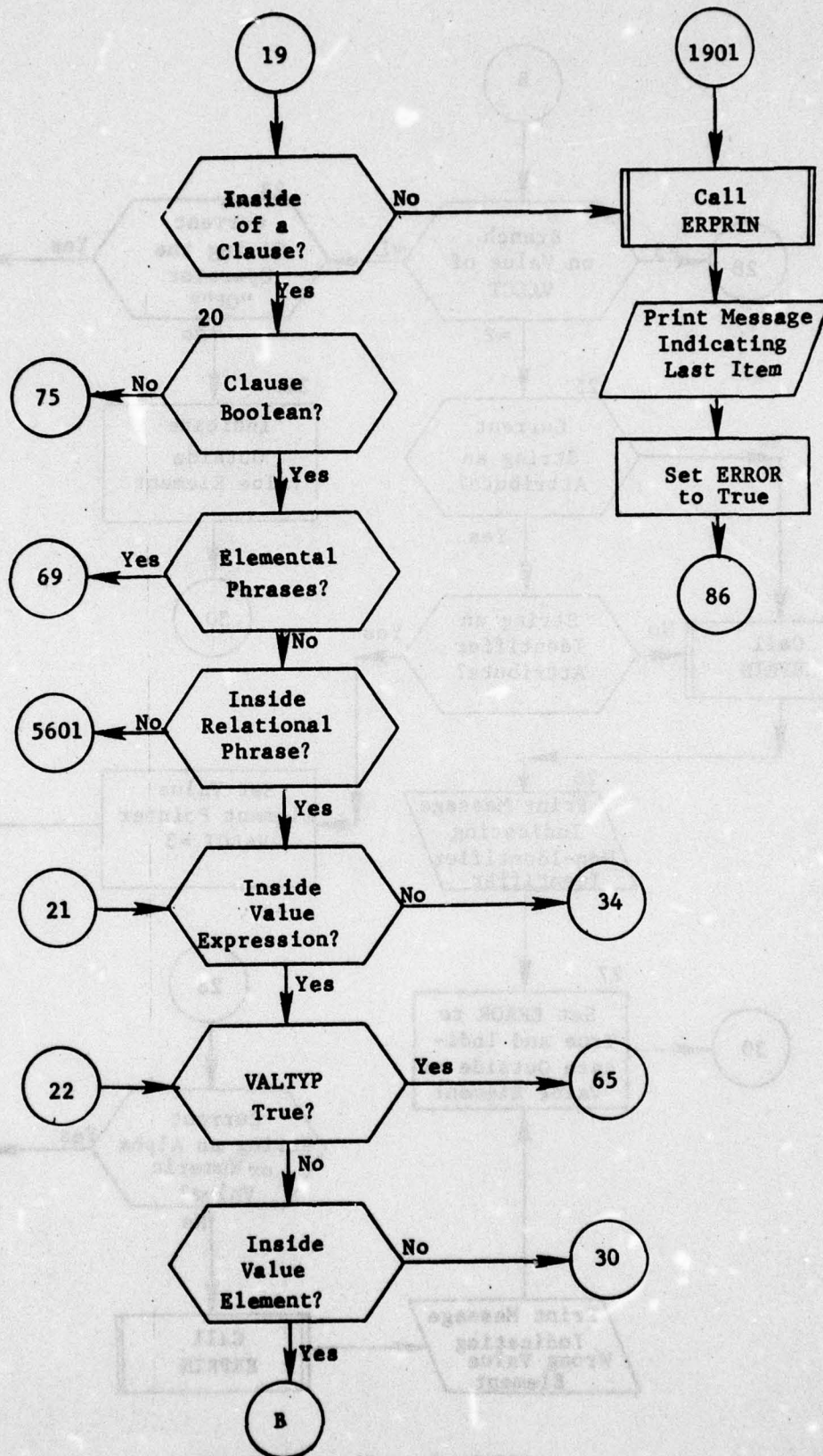


Figure 30. (Part 5 of 22)

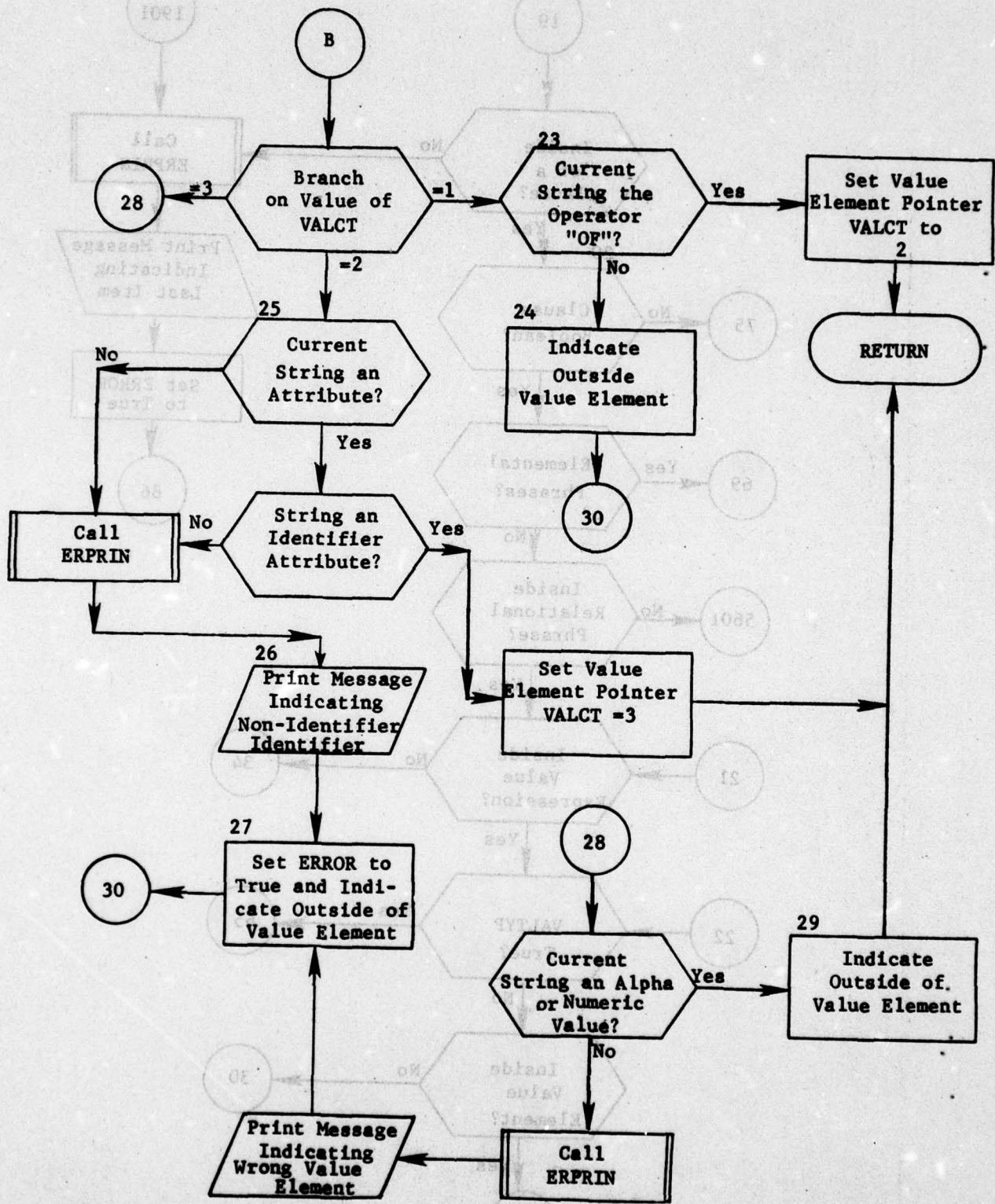


Figure 30. (Part 6 of 22)

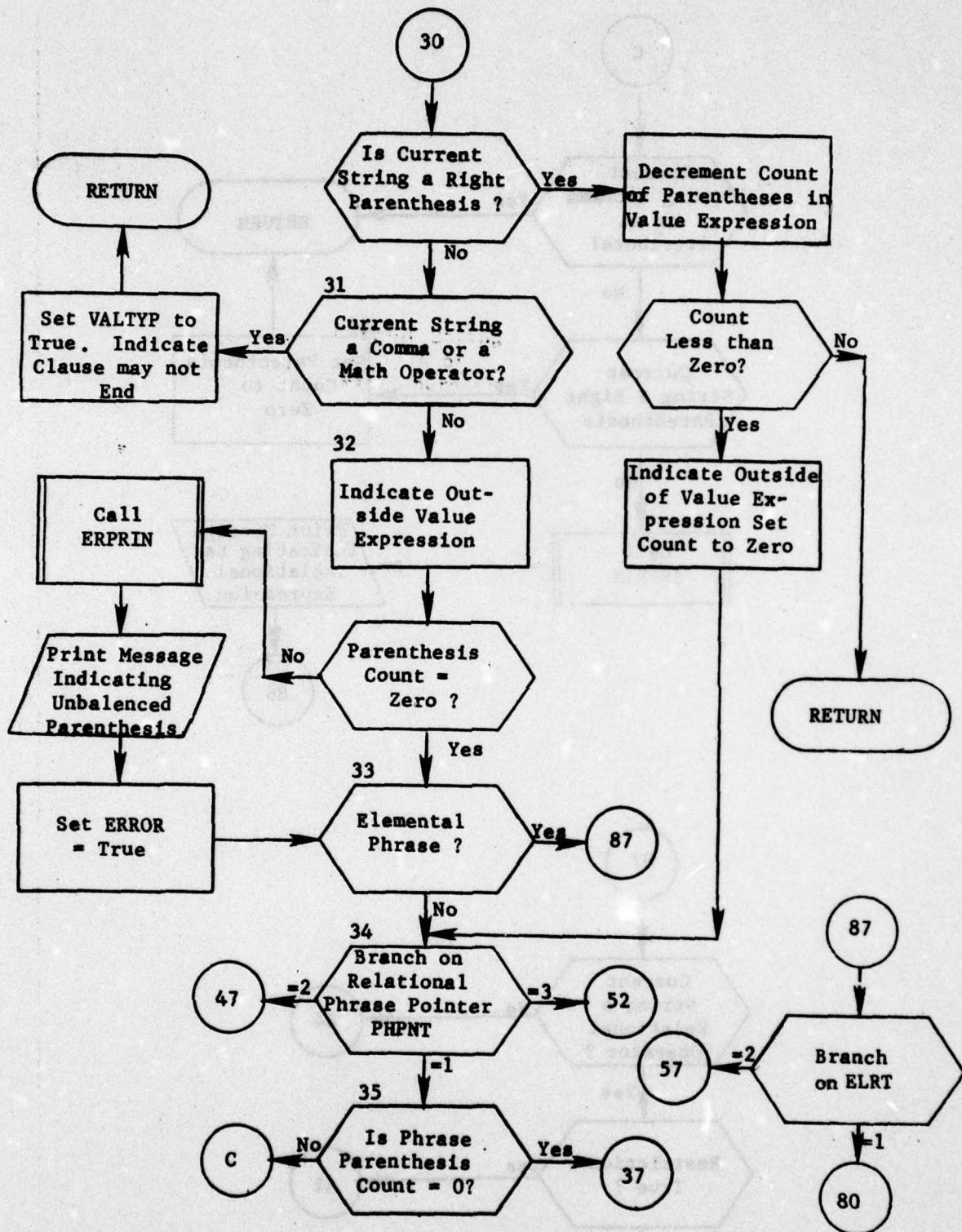


Figure 30. (Part 7 of 22)

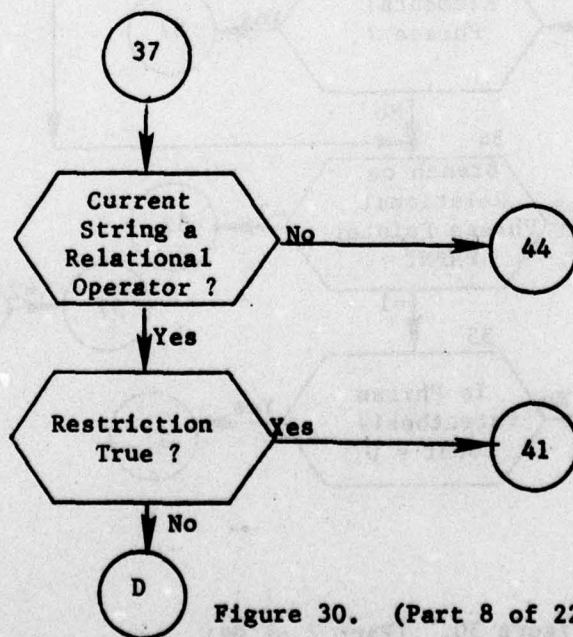
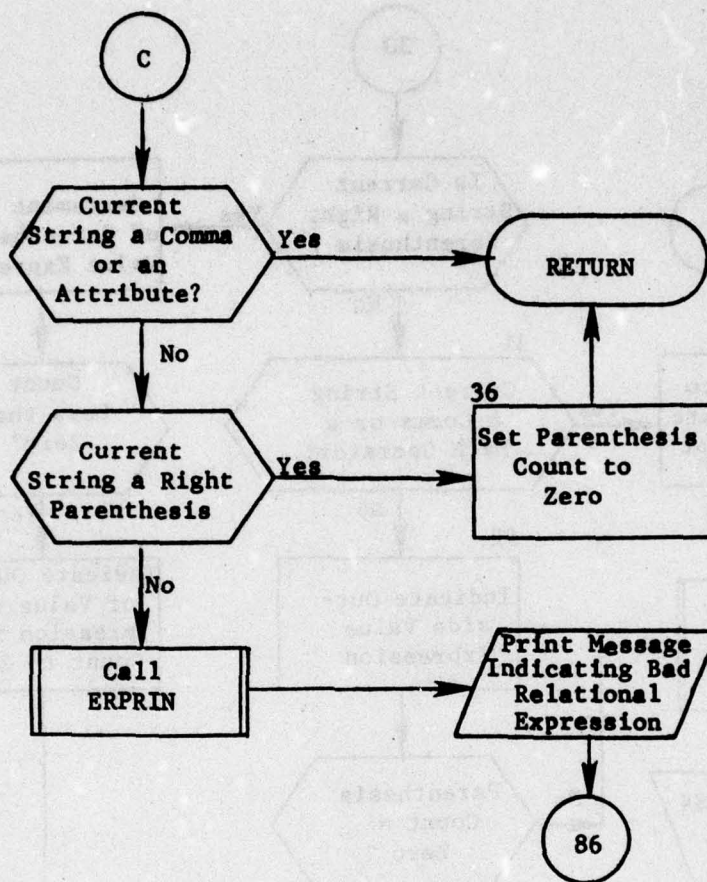


Figure 30. (Part 8 of 22)

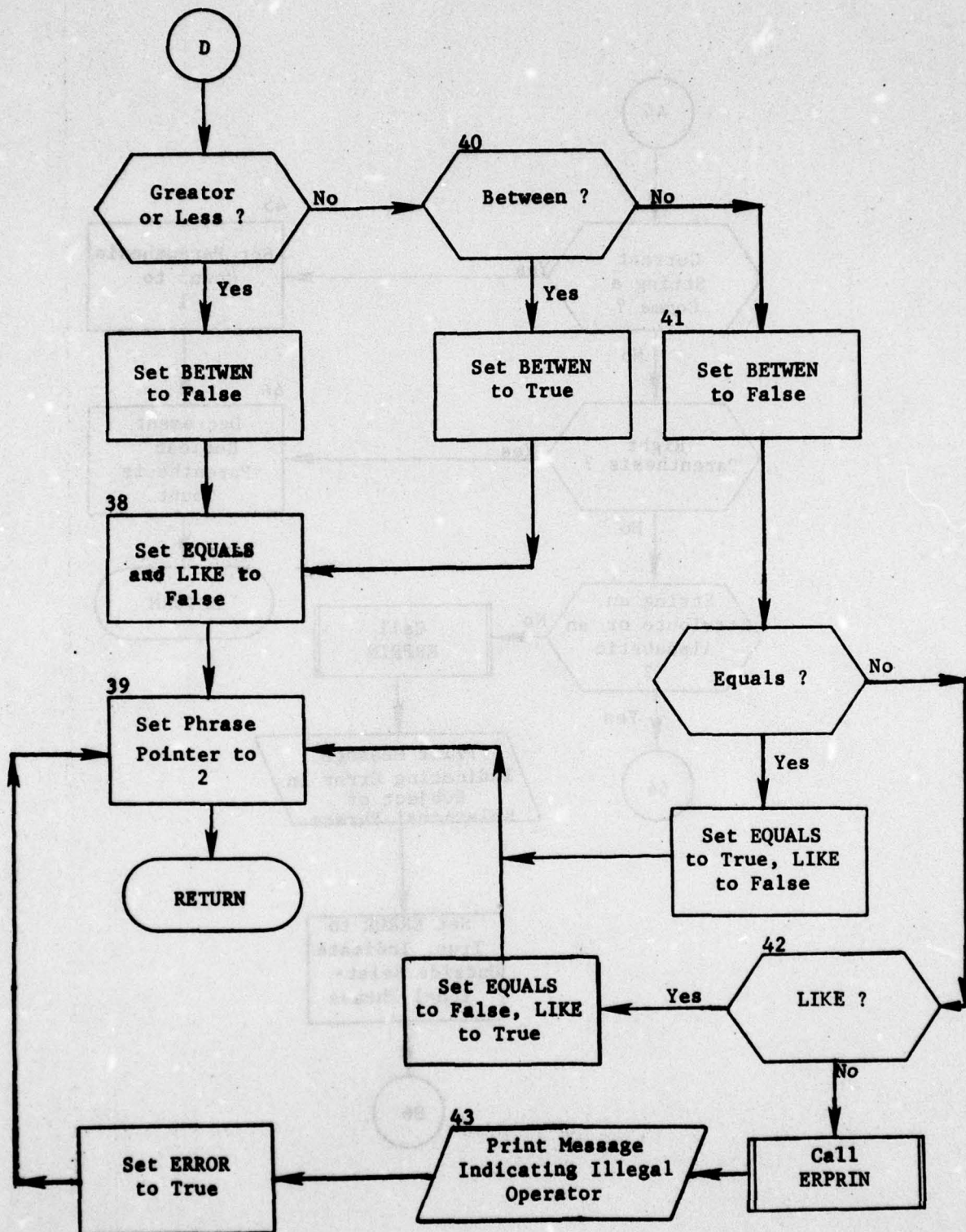


Figure 30. (Part 9 of 22)

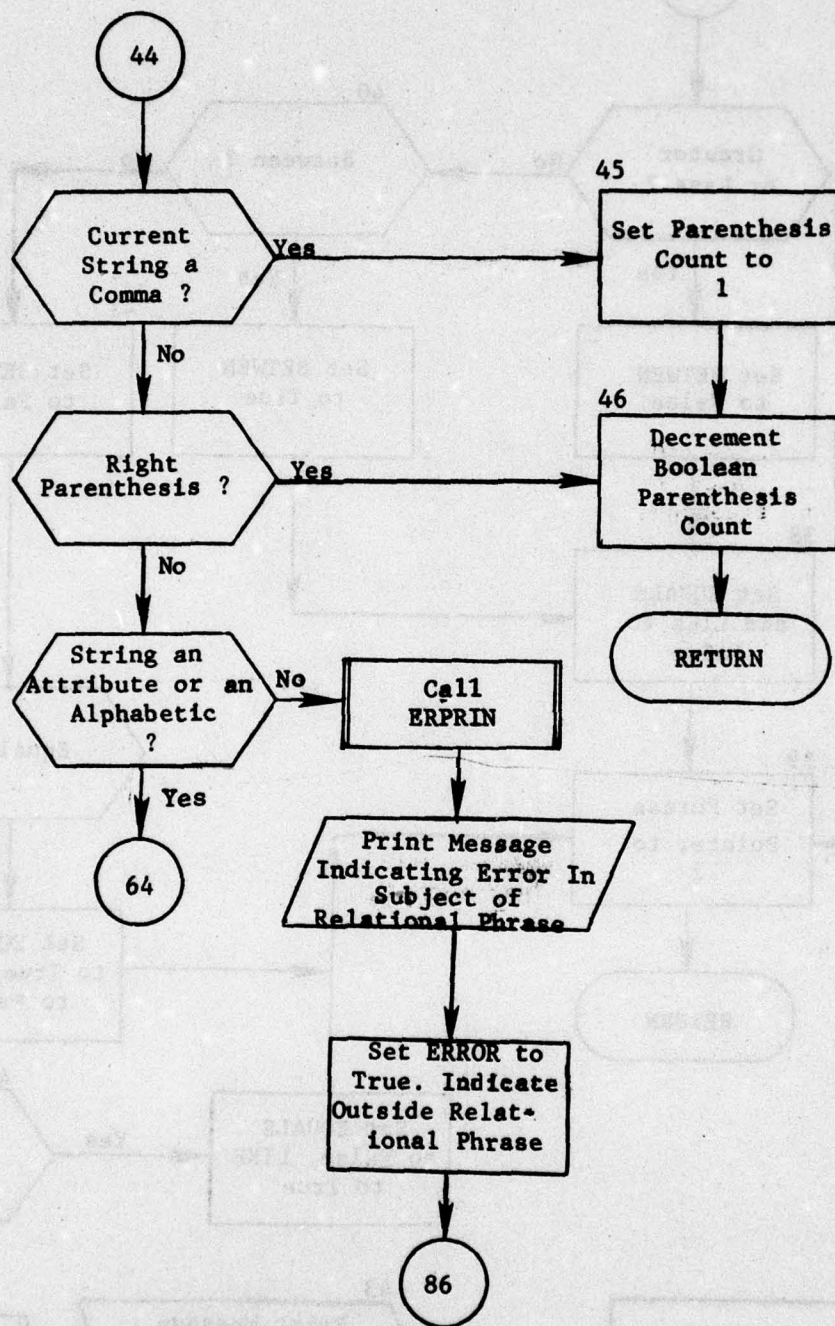


Figure 30. (Part 10 of 22)

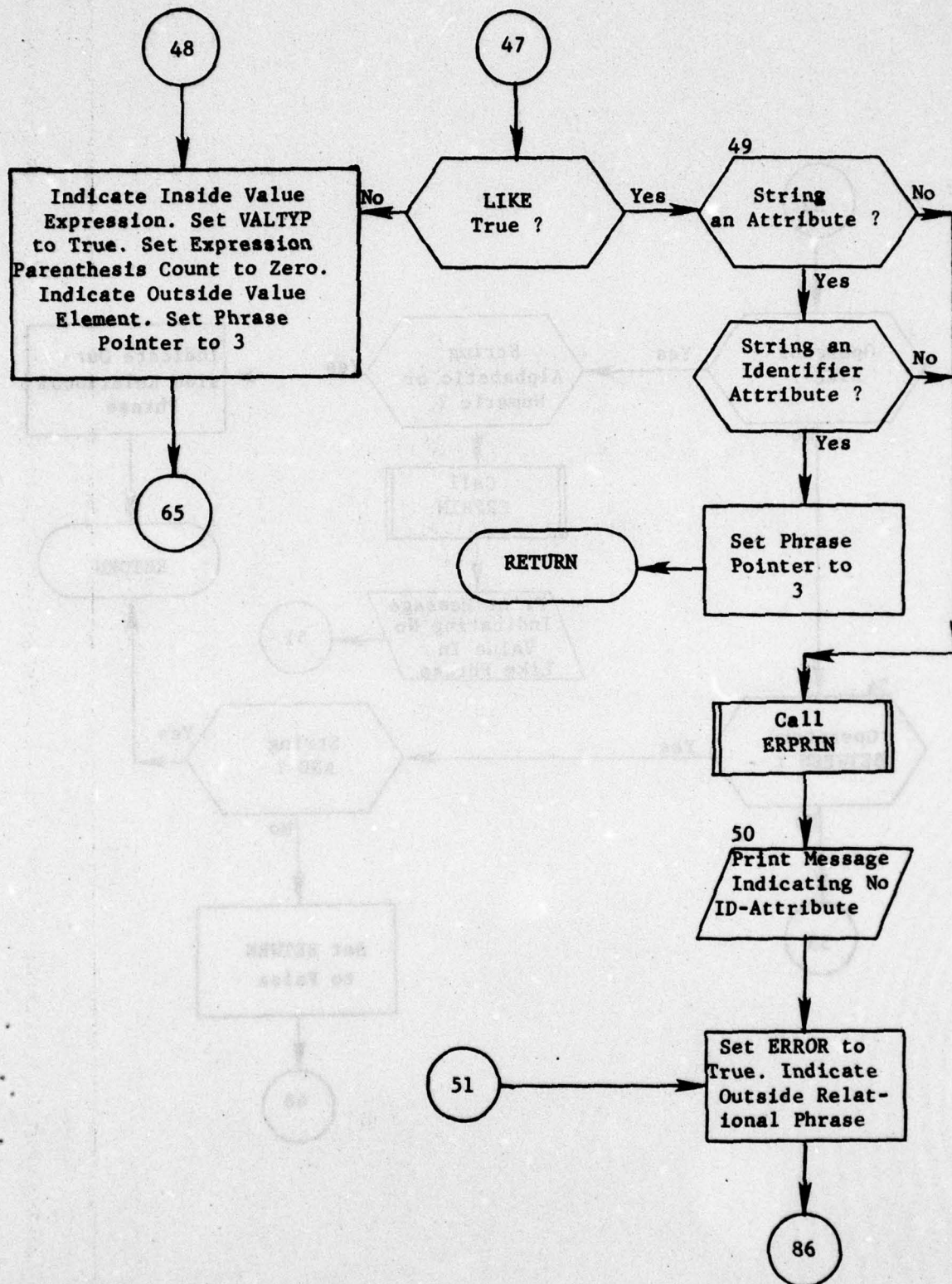


Figure 30. (Part 11 of 22)

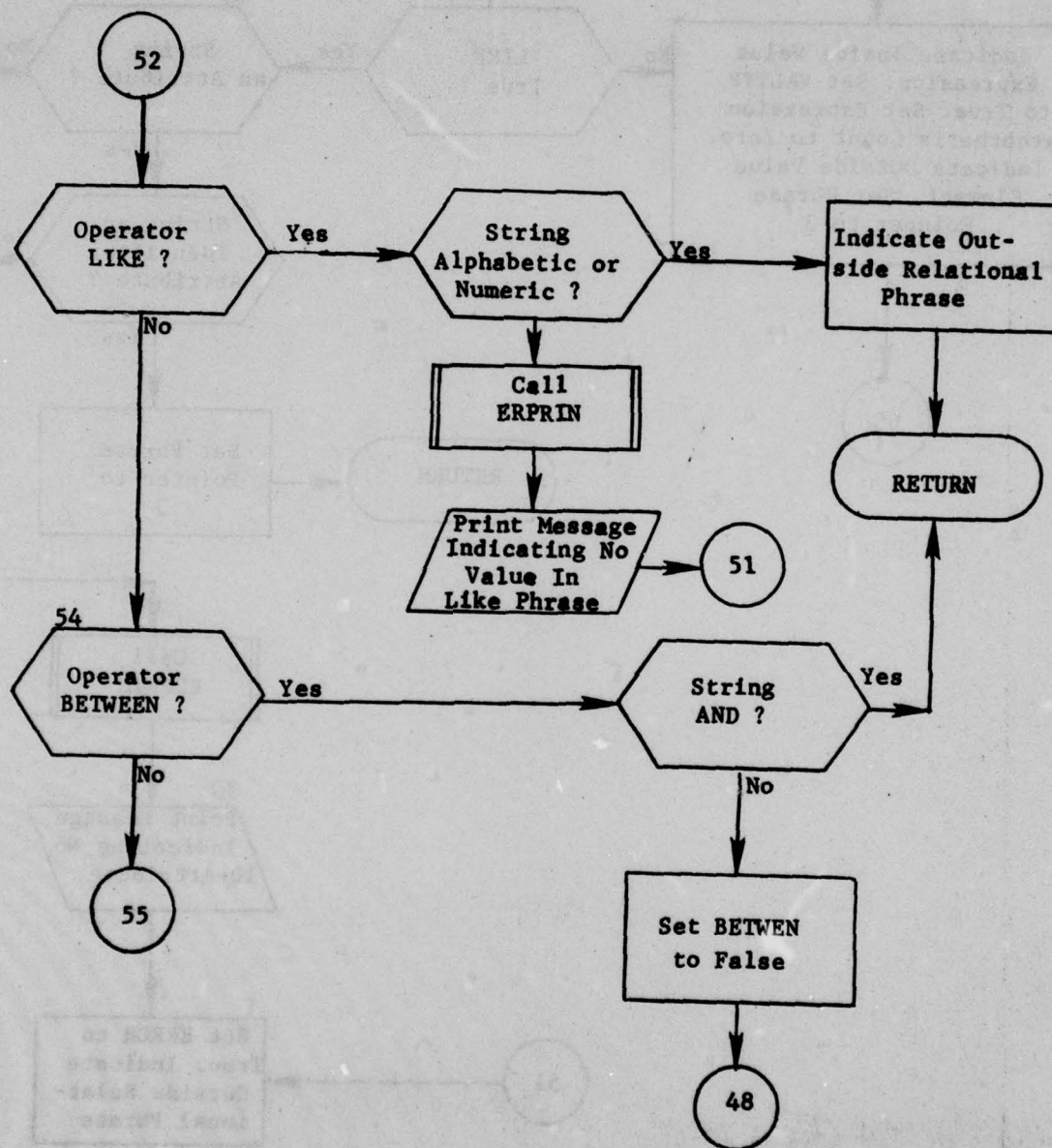


Figure 30. (Part 12 of 22)

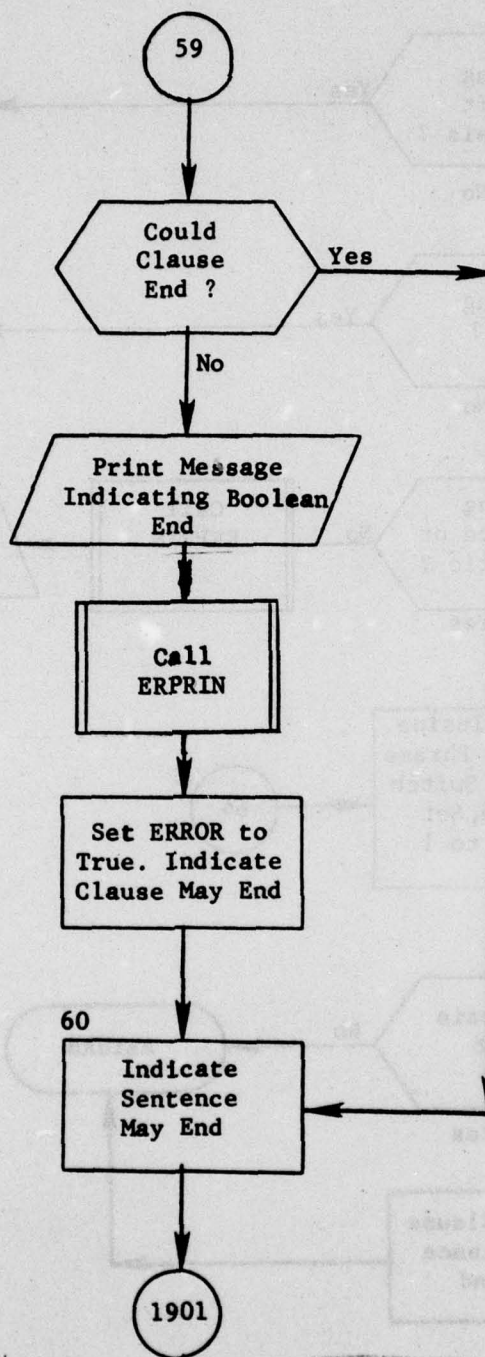


Figure 30. (Part 15 of 22)

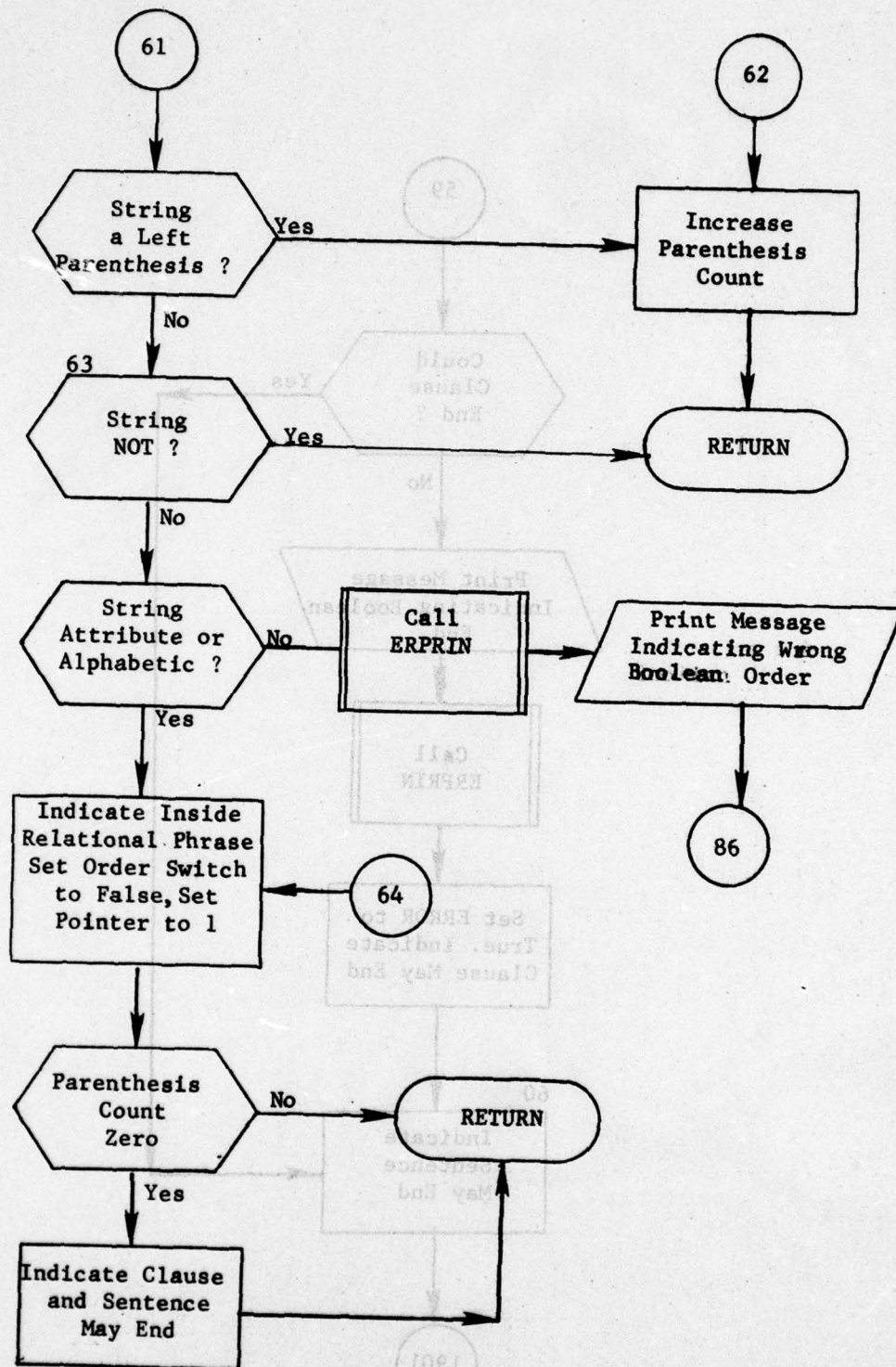


Figure 30. (Part 16 of 22)

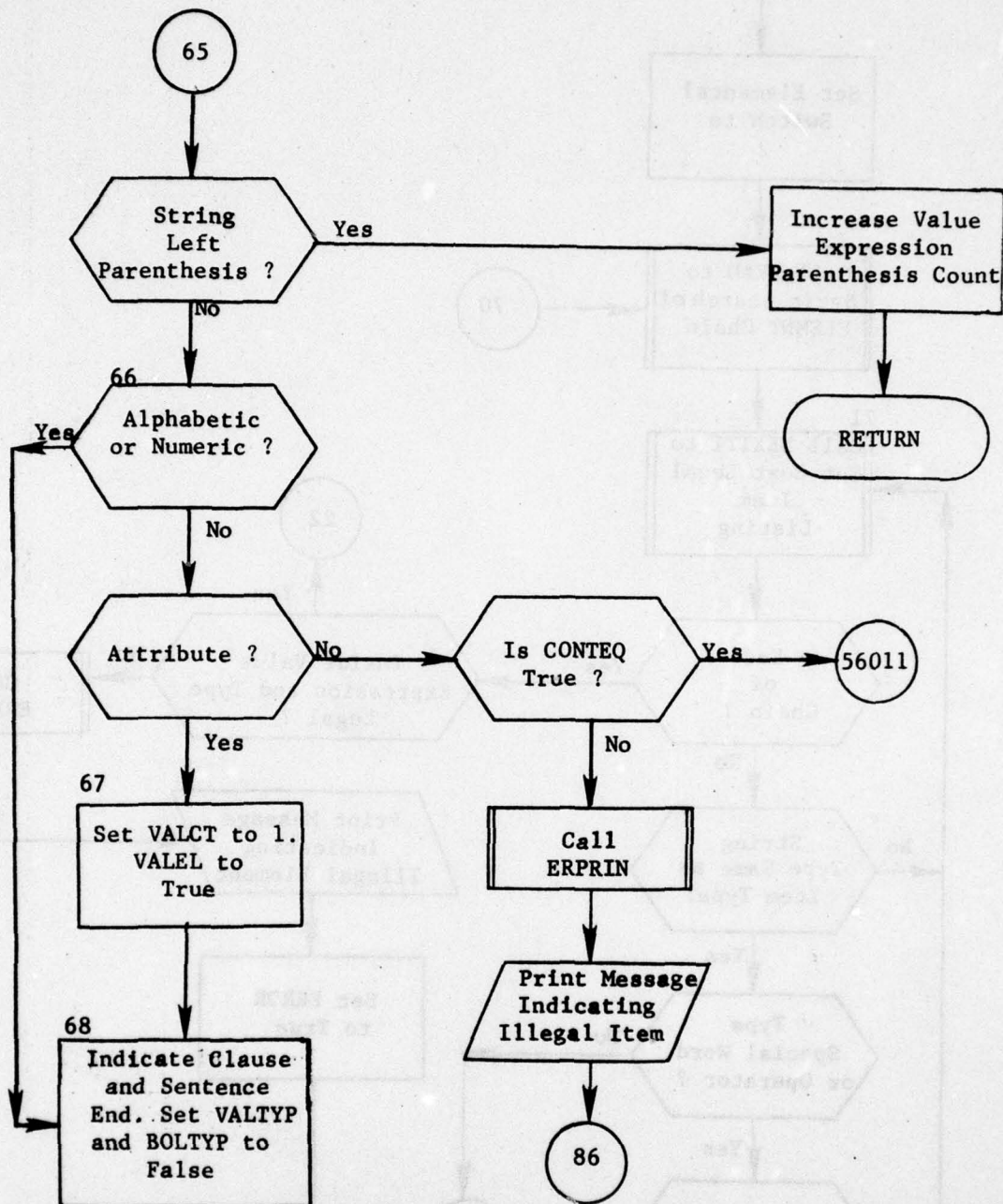


Figure 30. (Part 17 of 22)

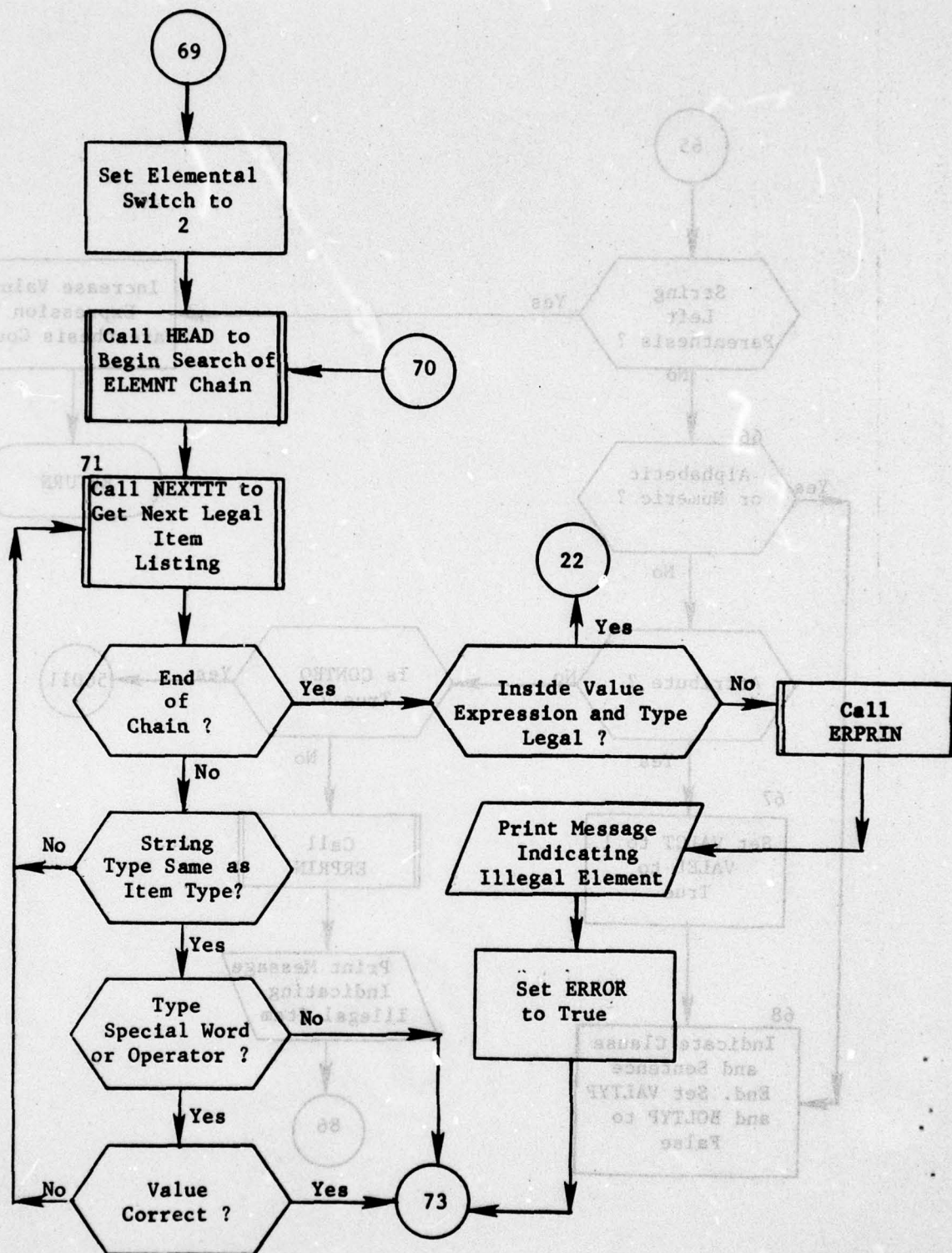


Figure 30. (Part 18 of 22)

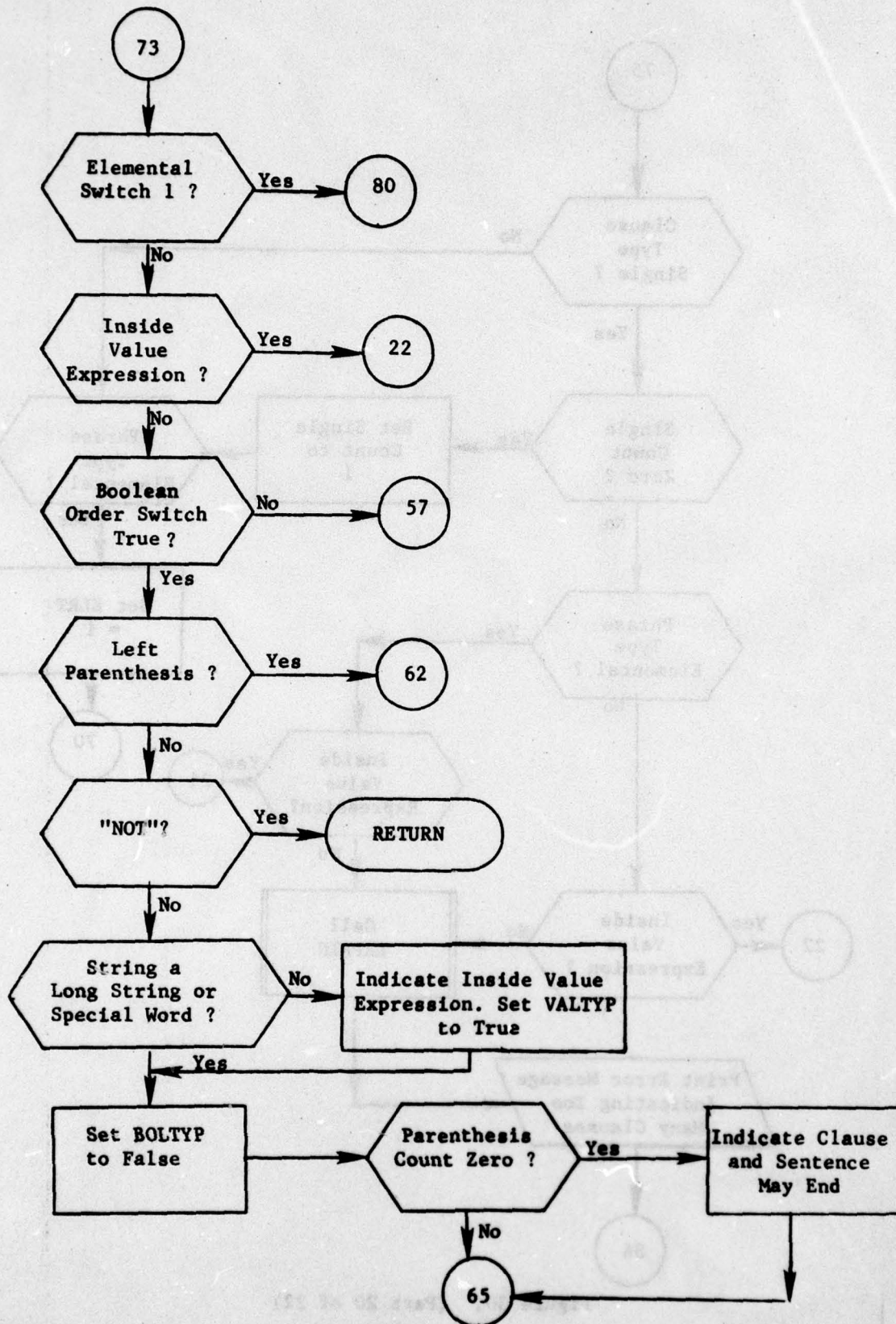


Figure 30. (Part 19 of 22)

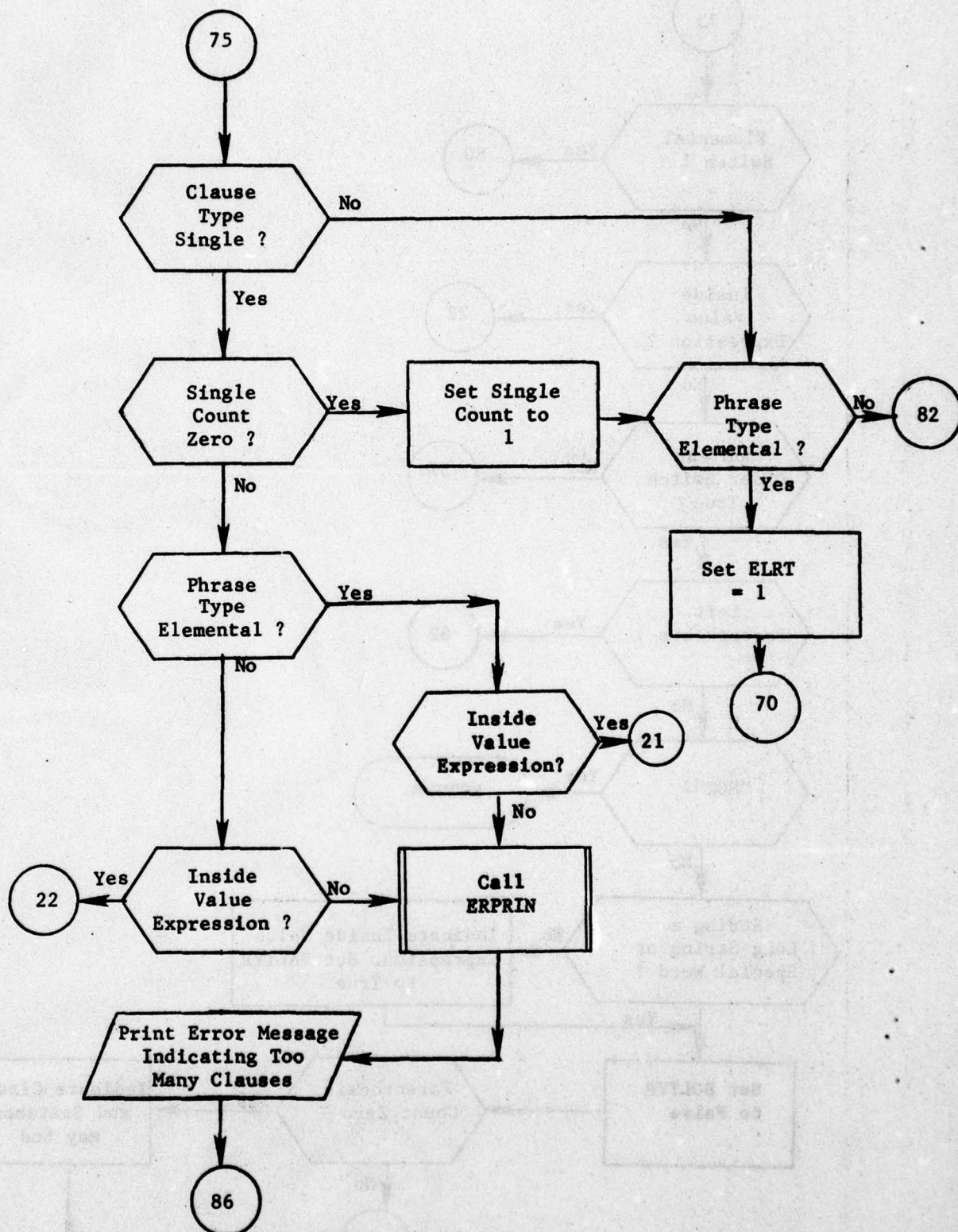


Figure 30. (Part 20 of 22)

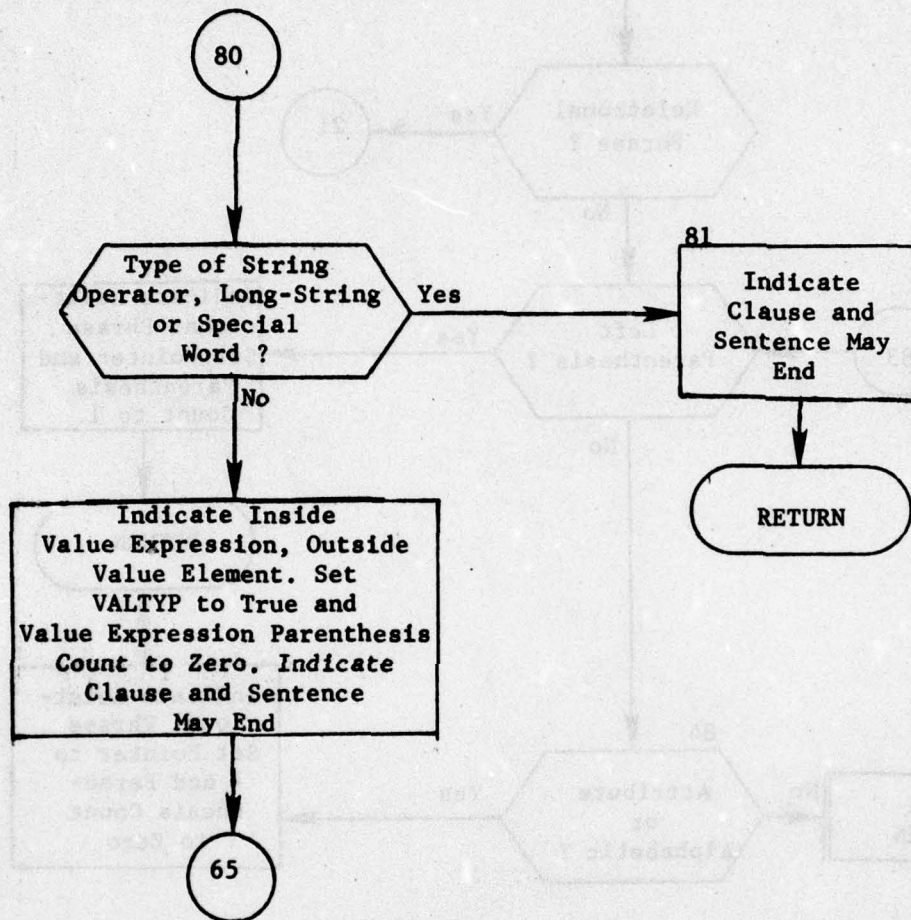


Figure 30. (Part 21 of 22)

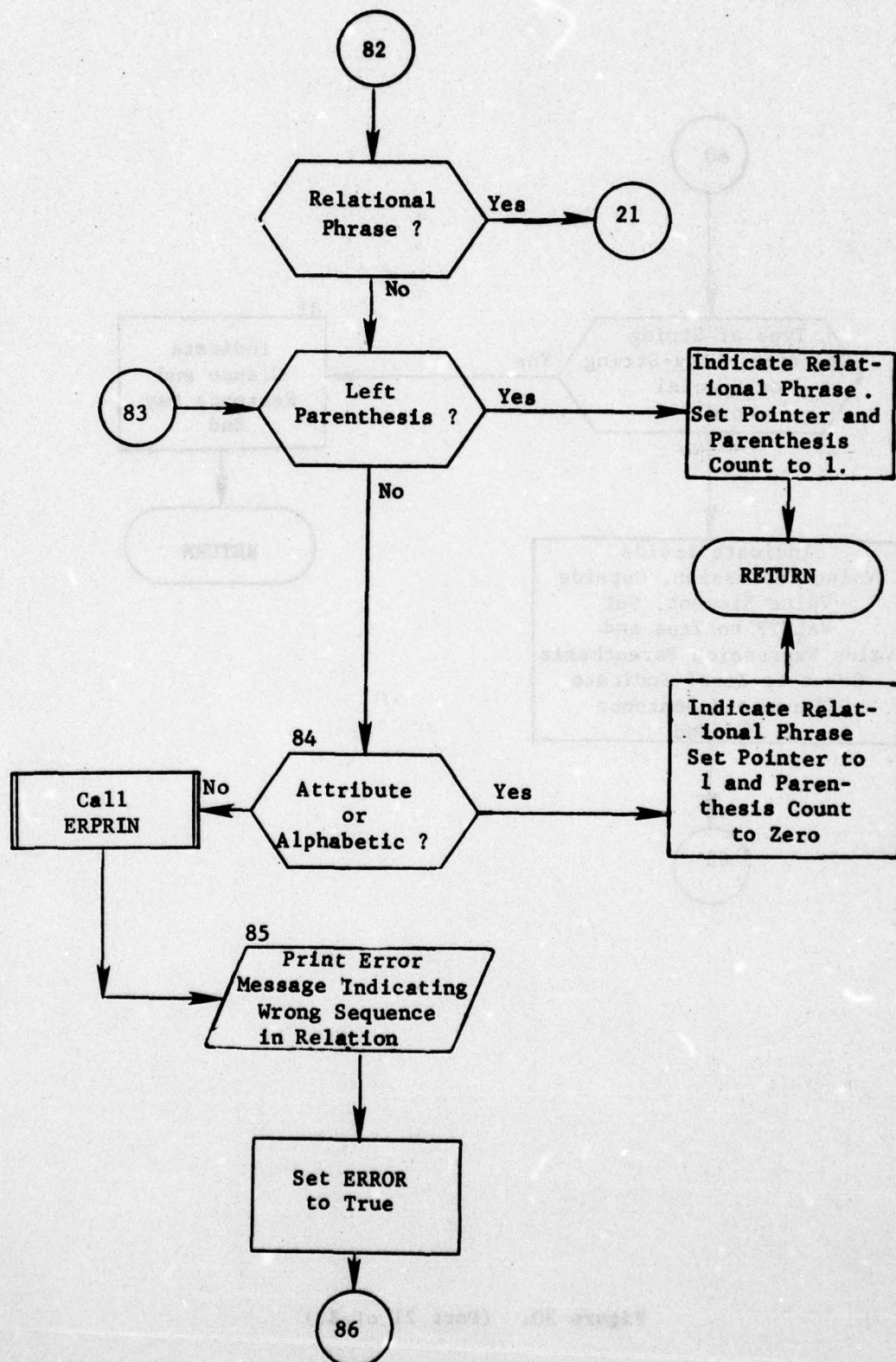


Figure 30. (Part 22 of 22)

SECTION 5. EDITDB MODULE

5.1 General Purpose

The EDITDB module performs two similar but quite distinct functions. First, it performs the standard consistency checks as outlined in table 13 in Users Manual, Volume I. Second, it will perform any set of non-standard edits desired by the user. This second function allows the user to check the validity of data attributes not covered by the standard edits or make certain that attributes fall within any desired limits other than those listed in the directory.

5.2 Input

The input to EDIT consists of the cards containing the generalized text english commands as described in the Users Manual. The data base can be in any stage completion after it has been initialized.

5.3 Output

EDIT will create no tapes or new data base records.

5.4 Concept of Operation

The execution of EDIT always performs consistency checks and determines counts of various subsections of the data base to ensure that limitations as dictated by the QUICK system are not exceeded. In addition, EDIT performs nonstandard checks as introduced by the user through the WHERE, FIELDS or WITH clauses. These generalized clauses allow for special subsections of the data base to be checked against ranges for validity as directed.

EDIT consists of four overlays each of which conducts a special function. The main objectives are met within these overlays.

5.5 Identification of Subroutine Functions

5.5.1 Subroutine COUNTS. This subroutine (or overlay) queries major portions of the defined data base and produces summary counts of various collections contained within the data. The processors to be executed within the QUICK system, uses local dimensioned arrays for efficient processing and, hence, certain limitations must be met. This subroutine, then, informs the user if certain subsections (geography, weapon types, and groups, etc.) exceeds any of these constraints. The code counts entries in a straightforward fashion.

5.5.2 Subroutine GENEDIT. This subroutine (or overlay link) is executed only if adverbs other than ONPRINTS appears within the input command. Its function is purely to drive the other subroutines within overlay link as directed by the type of input clause.

5.5.3 Subroutine BUILDTAB. The edit scheme is constructed within this subroutine. An edit scheme is a set of instructions that informs necessary subroutines how (or in what manner) the data base is to be queried. The philosophy behind this scheme is similar to that outlined for retrieval search schemes (see section 4.4).

5.5.4 Subroutine NORMAL. This subroutine is the third overlay link within EDIT and its function is simply to load default edit schemes.

5.5.5 Subroutine PROCEDIT. Edit schemes, either default or nonstandard, are executed by this subroutine (or overlay).

5.6 Edit Internal Common Blocks

The internal common blocks defined for this module are outlined in table 17.

DISTRIBUTION

<u>Addressee</u>	<u>Copies</u>
CCTC Codes	
Technical Library (C124)	3
C124 (Stock)	6
C313	1
C314	17
C600	1
 DCA Code	
205	1
 EXTERNAL	
Chief, Studies, Analysis and Gaming Agency, OJCS ATTN: SFD, Room 1D957, Pentagon, Washington, DC 20301	2
Chief of Naval Operations, ATTN: OP-96C4, Room 4A478, Pentagon, Washington, DC 20350	2
Commander-in-Chief, North American Air Defense Command ATTN: NPXYA, Ent Air Force Base, CO 80912	2
U. S. Air Force Weapons Laboratory (AFSC) ATTN: AFWL/SUL (Technical Library), Kirtland Air Force Base, NM 87117	1
Director, Strategic Target Planning, ATTN: (JPS), Offutt Air Force Base, NE 68113	2
Defense Documentation Center, Cameron Station, Alexandria, VA 22314	<u>12</u> 50

CONTENTS

Part I

Section	Page
ACKNOWLEDGMENT.....	ii
ABSTRACT.....	xi
1. GENERAL.....	1
2. INTEGRATED DATA BASE.....	9
3. CENTRAL OPERATIONS PROCESSOR.....	57
4. DATA MODULE.....	247
5. EDITDB MODULE.....	383

Part II

6. REPORT MODULE.....	435
6.1 Purpose.....	435
6.2 Input.....	435
6.3 Output.....	435
6.4 Concept of Operation.....	435
6.4.1 Print Scheme.....	438
6.4.2 Utility Tables.....	439
6.5 Identification of Subroutine Functions.....	439
6.5.1 Subroutine DESIGN.....	439
6.5.2 Subroutine ALTER.....	439
6.5.3 Subroutine DSPMAK.....	439
6.5.4 Subroutine PRINCE.....	439
6.6 Common Blocks.....	445
6.7 Subroutine ENTMOD.....	450
6.7.1 Subroutine DSPPUT.....	453
6.7.2 Subroutine TABMNT.....	456
6.8 Subroutine ALTER.....	460
6.9 Subroutine DESIGN.....	492
6.10 Subroutine DSPMAK.....	511
6.11 Subroutine PRINCE.....	547
(Entry DSPPUT)	
(Entry DSPGET)	
6.11.1 Subroutine XDEFN.....	568
6.11.2 Subroutine PRNATD.....	572.1
7. SAVE AND RESTORE MODULE (SRM).....	573
7.1 Purpose.....	573
7.2 Input.....	573

Section	Page
7.3 Output.....	573
7.4 Concept of Operation.....	573
7.5 Identification of Subroutine Functions.....	573
7.6 Common Blocks.....	573
7.7 Subroutine ENTMOD.....	575
8. EXTERNAL INTERFACE MODULE (EIM).....	581
8.1 Purpose.....	581
8.2 Input.....	581
8.3 Output.....	581
8.4 Concept of Operation.....	581
8.5 Identification of Subroutine Functions.....	581
8.5.1 Subroutine SIDAC.....	581
8.5.2 Subroutine TABLE.....	589
8.5.3 Subroutine BLDOTH.....	589
8.5.4 Subroutine PLOTDATA.....	589
8.5.5 Subroutine PLOTIT.....	589
8.6 Common Blocks.....	589
8.7 Subroutine ENTMOD.....	594
8.7.1 Subroutine CONVLL.....	597
8.8 Subroutine BLDOTH.....	599
8.8.1 Subroutine XEDEFN.....	628
8.9 Subroutine PLOTDATA.....	633
8.9.1 Subroutine PICS.....	651
8.9.2 Subroutine PROJCT.....	655
(Entry PROJCT)	
(Entry PROJ2)	
8.10 Subroutine SIDAC.....	658
8.11 Subroutine TABBLE.....	662
8.12 Subroutine PLOTIT.....	676
8.12.1 Subroutine FNDST.....	676.5
8.12.2 Subroutine INTRPL.....	676.8
8.12.3 Subroutine PLBLOFF.....	676.10
8.12.4 Subroutine PLOTINIT.....	676.14
8.12.5 Subroutine SUBPLOT.....	676.19
8.12.6 Subroutine SUBREAD.....	676.34
9. GENERAL UTILITIES.....	677
9.1 Purpose.....	677
9.2 Subroutine ABORT.....	683
9.3 Subroutine ATFNDR.....	685
9.4 Function ATN2PI.....	693
9.5 Subroutine CINSGET.....	695
9.6 Function DELLONG.....	697
9.7 Function DIFFLONG.....	699
9.8 Function DISTF.....	701
9.9 Subroutine DOTLINE.....	703

Section	Page
9.10 Subroutine FINDCLAS.....	705
9.11 Subroutine FINDSIDE.....	707
9.12 Subroutine FORMAK.....	709
9.13 Function GETCLOCK.....	711
9.14 Function GETDATE.....	713
9.15 Subroutine GETNXT.....	715
9.16 Subroutine GETSTR.....	719
9.17 Subroutine GETTAR.....	726
(Entry GETTAR)	
(Entries FNDTAR and GETDES)	
9.18 Function GLOG.....	737
9.19 Subroutine HOUSKEEP.....	739
9.20 Function IGET.....	741
9.21 Function IGETHOB.....	743
9.22 Subroutine INTERP.....	745
9.23 Subroutine INTRPGC.....	748
9.24 Subroutine INTPIECE.....	753
9.25 Subroutine IORFL.....	755
9.26 Subroutine IPUT.....	757
9.27 Subroutine ISOFF.....	759
9.28 Function ITLE.....	761
9.29 Function IWANT.....	763
9.30 Function KEYMAKE.....	765
9.31 Function LATBT.....	767
9.32 Subroutine LINKUP.....	769
9.33 Subroutine LREORDER.....	779
9.34 Subroutine MAPEDGE.....	781
9.35 Function NUMGET.....	783
9.36 Subroutine OFVAL.....	786
9.37 Subroutine ORDER.....	790
9.38 Subroutine PAGESKP.....	792
9.39 Subroutine PIECEIT.....	794
9.40 Subroutine PIECENUM.....	799
9.41 Subroutine PLTTIC.....	801
9.42 Subroutine PRIMHD.....	803
9.43 Subroutine PSREC.....	805
(Entry PSREC)	
(Entry PSPUT)	
(Entry PSRWD)	
(Entries XSORT and PSNXT)	
9.44 Subroutine REORDER.....	811
9.45 Function RLBRT.....	814
9.46 Subroutine SETORD.....	816
(Entry SETORD)	
(Entry RESORD)	
9.47 Subroutine SETSCH.....	820
9.48 Function SLOG.....	830
9.49 Subroutine SORTIT.....	832
9.50 Function SSKPC.....	836

Section	Page
9.51 Function STD.....	840
9.52 Subroutine SVTP.....	842
(Entry FILLOD)	
(Entry FILSAV)	
9.53 Subroutine TICMAKE.....	848
9.54 Function TIMEDAY.....	851
9.55 Subroutine TIMEME.....	853
9.56 Function TOFM.....	856
9.57 Subroutine UNCODE.....	858
9.58 Subroutine VALFND.....	860
9.59 Function VALTAR.....	870
9.60 Function XCOORD.....	872
9.61 Subroutine XMATH.....	874
9.62 Subroutine XWHERE.....	878
9.63 Function ZTAN.....	888
APPENDIXES	
A. COP External Common Blocks.....	891
B. Executable Job Control Language (JCL) QUICK System....	895
C. PERFORM Program	909
DISTRIBUTION.....	
DD Form 1473.....	925

ILLUSTRATIONS (PART II)

Figure		Page
79	Subroutine DESIGN Macro Flow.....	440
80	Subroutine ALTER Macro Flow.....	441
81	Subroutine DSPMAK Macro Flow.....	443
82	Subroutine PRINCE Macro Flow.....	446
83	Subroutine ENTMOD (REPORT).....	451
84	Subroutine DSPPUT.....	454
85	Subroutine TABMNT.....	457
86	Subroutine ALTER: Step One.....	461
87	Subroutine ALTER: Step Two.....	464
88	Subroutine ALTER: Step Three.....	467
89	Subroutine ALTER: Step Four.....	470
90	Subroutine ALTER: Step Five.....	478
91	Subroutine ALTER: Step Six.....	480
92	Subroutine ALTER: Step Seven.....	491
93	Subroutine DESIGN: Step One.....	493
94	Subroutine DESIGN: Step Two.....	496
95	Subroutine DESIGN: Step Three.....	498
96	Subroutine DESIGN: Step Four.....	501
97	Subroutine DESIGN: Step Five.....	503
98	Subroutine DESIGN: Step Six.....	506
99	Subroutine DSPMAK: Step One.....	512
100	Subroutine DSPMAK: Step Two.....	514
101	Subroutine DSPMAK: Step Three.....	516
102	Subroutine DSPMAK: Step Four.....	518
103	Subroutine DSPMAK: Step Five.....	520
104	Subroutine DSPMAK: Step Six.....	521
105	Subroutine DSPMAK: Step Seven.....	526
106	Subroutine DSPMAK: Step Eight.....	528
107	Subroutine DSPMAK: Step Nine.....	531
108	Subroutine DSPMAK: Step Ten.....	543
109	Subroutine PRINCE: Step One.....	548
110	Subroutine PRINCE: Step Two.....	551
111	Subroutine PRINCE: Step Three.....	556
112	Subroutine PRINCE: Step Four.....	558
113	Subroutine PRINCE: Step Five.....	560
114	Subroutine XDEFN.....	569
114.1	Subroutine PRNATD.....	572.2
115	Subroutine ENTMOD (SRM).....	576
116	Subroutine BLDOTH: Macro Flow.....	590
117	Subroutine ENTMOD (EIM).....	595
118	Subroutine CONVLL.....	598
119	Subroutine BLDOTH: Step One.....	600
120	Subroutine BLDOTH: Step Two.....	607
121	Subroutine BLDOTH: Step Three.....	612
122	Subroutine BLDOTH: Step Four.....	614
123	Subroutine BLDOTH: Step Five.....	616
124	Subroutine BLDOTH: Step Six.....	618

Figure

Page

125	Subroutine XEDEFN.....	629
126	Subroutine PLOTDATA.....	634
127	Subroutine PICS.....	652
128	Subroutine PROJCT.....	656
129	Subroutine SIDAC.....	659
130	Subroutine TABBLE.....	663
130.1	Subroutine PLOTIT.....	676.1
130.2	Subroutine FNDSRT.....	676.6
130.3	Subroutine INTRPL.....	676.9
130.4	Subroutine PLBLOFF.....	676.11
130.5	Subroutine PLOTINIT.....	676.15
130.6	Subroutine SUBPLOT.....	676.21
130.7	Subroutine SUBREAD.....	676.35
131	Subroutine ABORT.....	684
132	Subroutine ATFNDR.....	687
133	Function ATN2PI.....	694
134	Subroutine CINSGET.....	696
135	Function DELONG.....	698
136	Function DIFFLONG.....	700
137	Function DISTF.....	702
138	Subroutine DOTLINE.....	704
139	Subroutine FINDCLAS.....	706
140	Subroutine FINDSIDE.....	708
141	Subroutine FORMAK.....	710
142	Function GETCLOCK.....	712
143	Function GETDATE.....	714
144	Subroutine GETNXT.....	717
145	Subroutine GETSTR.....	720
146	Subroutine GETTAR.....	728
147	Function GLOG.....	738
148	Subroutine HOUSKEEP.....	740
149	Function IGET.....	742
150	Function IGETHOB.....	744
151	Subroutine INTERP.....	747
152	Coordinate System for INTRPGC.....	749
153	Subroutine INTRPGC.....	752
154	Subroutine INTPIECE.....	754
155	Subroutine IORFL.....	756
156	Subroutine IPUT.....	758
157	Subroutine ISOFF.....	760
158	Function ITLE.....	762
159	Function IWANT.....	764
160	Function KEYMAKE.....	766
161	Function LATBT.....	768
162	Subroutine LINKUP.....	770
163	Subroutine LREORDER.....	780
164	Subroutine MAPEDGE.....	782
165	Function NUMGET.....	785
166	Subroutine OFVAL.....	787

Figure		Page
167	Subroutine ORDER.....	791
168	Subroutine PAGESKP.....	793
169	Subroutine PIECEIT.....	796
170	Subroutine PIECENUM.....	800
171	Subroutine PLTTIC.....	802
172	Subroutine PRIMHD.....	804
173	Subroutine PSREC.....	807
174	Subroutine REORDER.....	812
175	Function RLBRT.....	815
176	Subroutine SETORD.....	818
177	Subroutine SETSCH.....	822
178	Function SLOG.....	831
179	Subroutine SORTIT.....	833
180	Function SSKPC.....	838
181	Function STD.....	841
182	Subroutine SVIP.....	843
183	Subroutine TICMAKE.....	849
184	Function TIMEDAY.....	852
185	Subroutine TIMEME.....	855
186	Focal Point Calculations.....	857.2
186.1	Function TOFM.....	857.5
187	Subroutine UNCODE.....	859
188	Subroutine VALFND.....	862
189	Function VALTAR.....	871
190	Function XCOORD.....	873
191	Subroutine XMATH.....	875
192	Subroutine XWHERE.....	879
193	Function ZTAN.....	889
194	H* Creation From Source.....	896
195	H* Creation From Object.....	902
196	Utility Library Creation.....	906
197	Program PERFORM.....	912
198	Subroutine READIN.....	921
199	Subroutine IDENT.....	921.2

ABSTRACT

The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, provide output summaries, and produce tapes to simulator subsystems external to QUICK. QUICK has been programmed in FORTRAN for use on the CCTC HIS 6000 computer system.

The QUICK Maintenance Manual consists of four volumes: Volume I, Data Management Subsystem; Volume II, Weapon/Target Identification Subsystem; Volume III, Weapon Allocation Subsystem, Volume IV, Sortie Generation Subsystem. The Maintenance Manual complements the other QUICK Computer System Manuals to facilitate application of the war gaming system. This volume, Volume I in two parts, provides the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the modules (programs) and subroutines of the Data Management subsystem. Companion documents are:

a. **USERS MANUAL**

Computer System Manual CSM UM 9-77, Volume I

Computer System Manual CSM UM 9-77, Volume II

Computer System Manual CSM UM 9-77, Volume III

Computer System Manual CSM UM 9-77, Volume IV

Provides detailed instructions for applications of the system

b. **TECHNICAL MEMORANDUM**

Technical Memorandum TM 153-77

Provides a nontechnical description of the system for senior management personnel

6.4.2 Utility Tables. The subroutine TABMNT creates, maintains, and deletes a set of utility tables. These are arrays which TABMNT can keep internally for a maximum of 100 words. If more space is required for the table, the current 100 words are moved into common block C40 and a TABLEZ record created. When data is desired from a particular table, an index is used to determine which set of 100 words contains the desired data. The appropriate TABLEZ record is retrieved if necessary and the data moved from C40 to internal storage. TABMNT can maintain up to five utility tables each of which can contain a maximum of 1,000 words. The tables are identified by a number from one to five (i.e., 'utility table 1'). Both DESIGN and ALTER use these tables to pass clauses to DSPMAK. PRINT uses the tables to build a new WHERE clause and to retain headers and print scheme.

6.5 Identification of Subroutine Functions

6.5.1 Subroutine DESIGN. This subroutine (or overlay) carries out the function of building a new set of display specifications from scratch. First the DISPLAY clause is read and the display table set for construction. The SETTING clause is used to set values. Each DEFINE is scanned for errors, and is stored in a table. The WHERE, SORT, and FORMAT clauses are also scanned for errors and saved (see figure 79).

6.5.2 Subroutine ALTER. This subroutine (or overlay) makes alterations to old displays or constructs a new display based upon an old one. First the DISPLAY clause is used to find the old display. Then each clause of the old display is either saved in the utility tables or replaced by a new clause which has been input. The old FORMAT clause may have portions modified according to user instructions. Finally, the new display table is set up for creation or the old one deleted and reset depending upon the DISPLAY clause option selected (see figure 80).

6.5.3 Subroutine DSPMAK. This subroutine takes the utility tables built by either the DESIGN or ALTER subroutines and uses them to construct a display. First all clauses are scanned for attributes. Then the attributes collected are used to build a retrieval scheme. (For details of this process see section 4.4.) DEFINES are placed in proper execution order and their execution tables (table 18, elements 2-7) are built. The WHERE clause is scanned for DEFINE variables and altered accordingly. The sort scheme and print scheme are now built. Finally, all the constructed elements are stored in the data base as a display table (see figure 81).

6.5.4 Subroutine PRINCE. This subroutine prints a specified display. First the DISPLAY clause is used to find the desired display table and the various schemes and tables are read from the display table. If the input display name is "DIRECTORY" the PRNATD subroutine is called to print the directory. If a new WHERE clause has been input it replaces

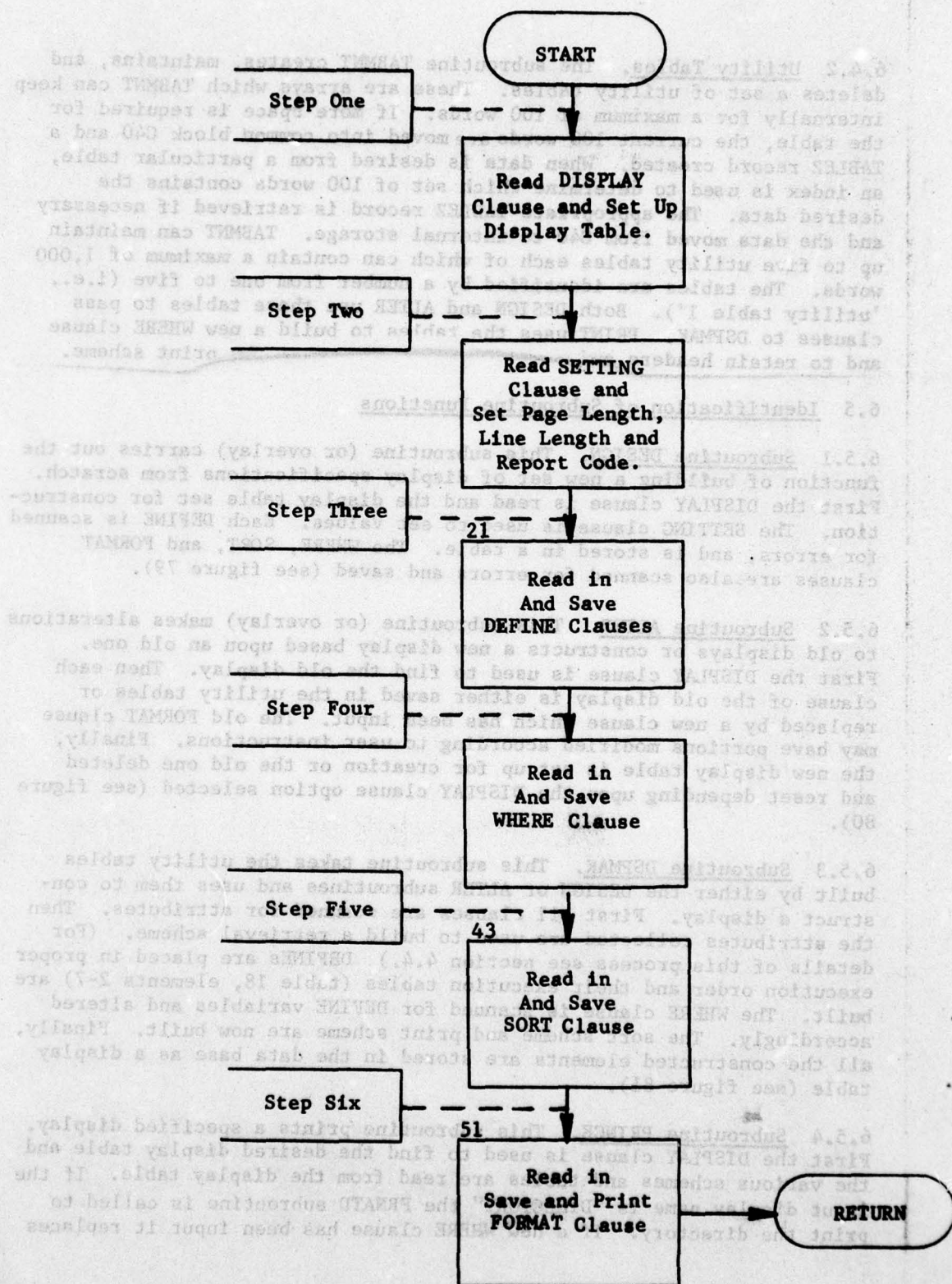
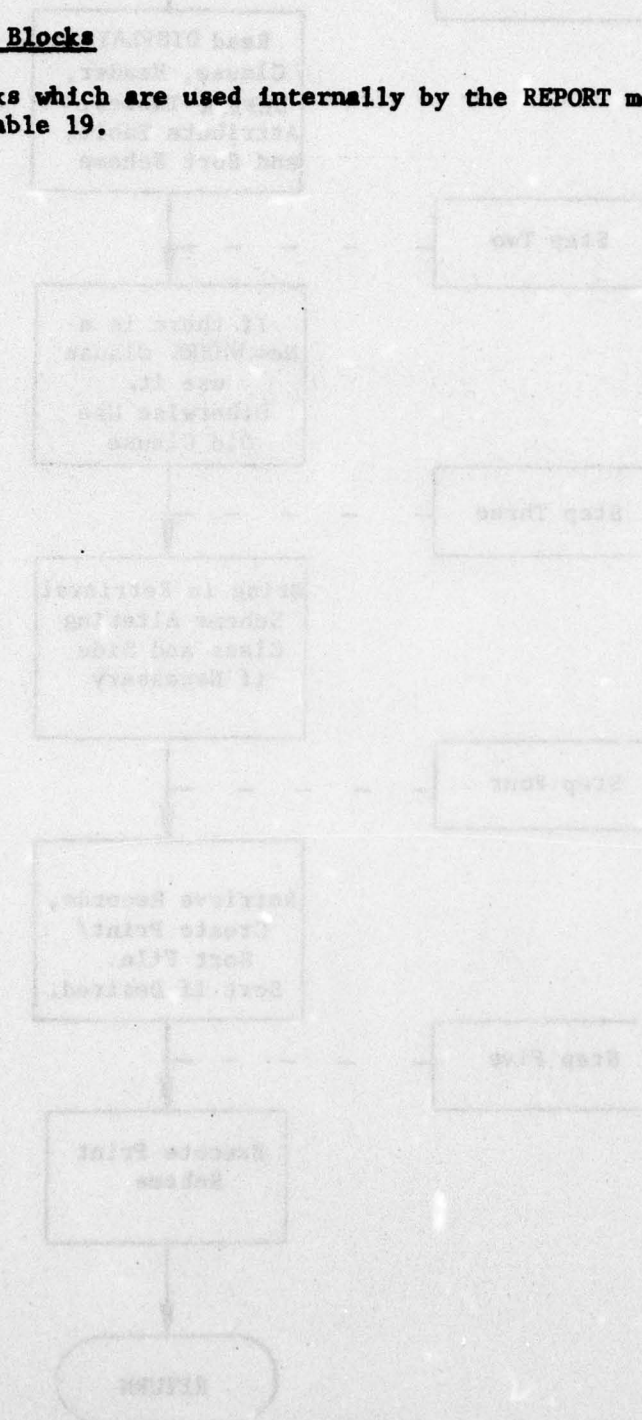


Figure 79. Subroutine DESIGN Macro Flow

| any old one. The retrieval scheme is used to build a file on the desired information. This file is then sorted if a sort has been requested. Finally, the print scheme is executed to produce the desired report (see figure 82).

6.6 Common Blocks

Common blocks which are used internally by the REPORT module are displayed in table 19.



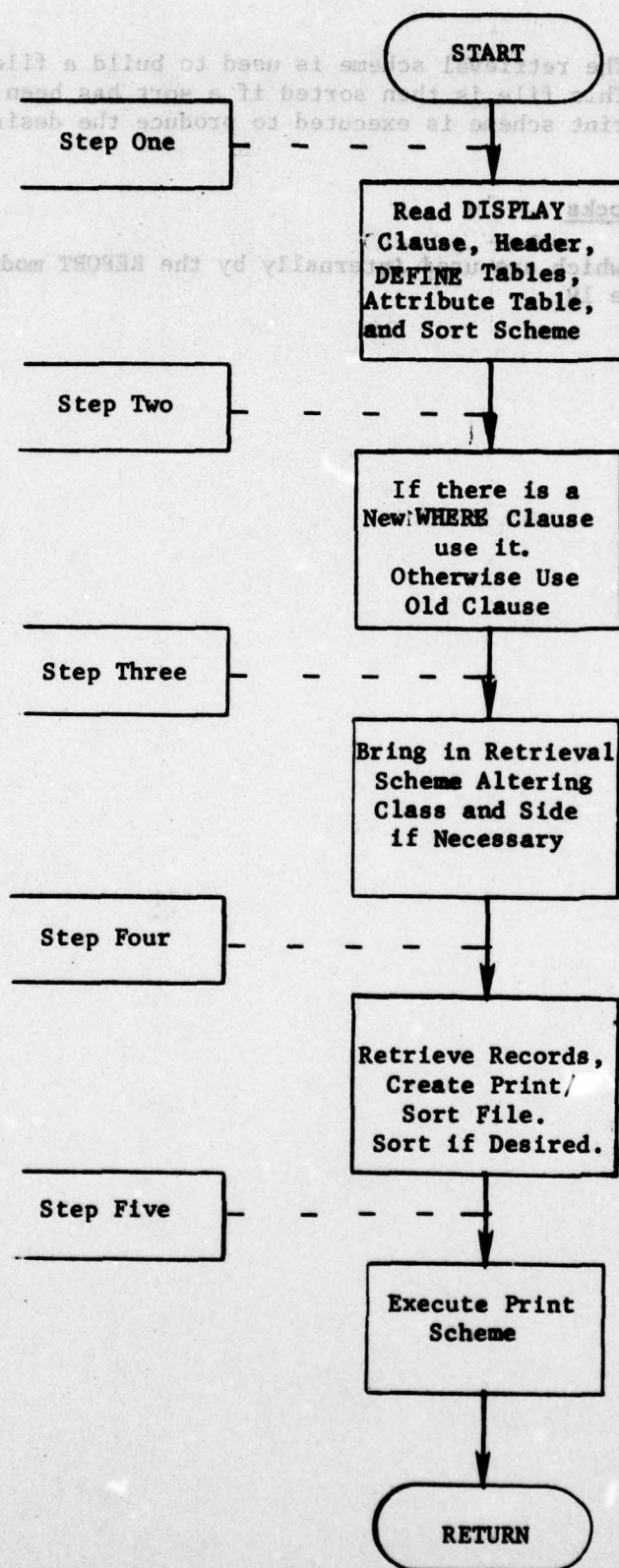


Figure 82. Subroutine PRINCE Macro Flow

6.11 Subroutine PRINCE*

PURPOSE: To print a display

ENTRY POINTS: PRINCE

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C20, C30, DEFVAR, DSPHED, IDPT, INS, NONPAG, OOPS, PAGPAG, PSCOM, SCHEME, SORSCH, ZEES

SUBROUTINES CALLED: DSPGET, GETNXT, HDFND, HEAD, INSFLS, INSGET, INSPUT, NEXTTT, PRNATD, PSNXT, PSPUT, PSREC, PSRWD, RETRV, TABMNT, UNCODE, XDEFN, XSORT, XWHERE

CALLED BY: ENTMOD (REPORT)

Method:

Step One

The input is scanned for the DISPLAY clause and the display name is found (if no DISPLAY clause is provided the name 'QTEMPORARYQQ' is used). The DISPLAY chain is searched for the appropriate display table. Next, subroutine DSPGET is called to retrieve the display table header, define tables (instructions are stored in utility table 1), attributes (instructions 1-70 in table 18) (see figure 109). If the display name is "DIRECTORY", subroutine PRNATD is called and processing ends.

Step Two

The input is now scanned for a WHERE clause. If none is found the old clause is retrieved by DSPGET and stored via INSPUT in the input instruction tables (this clause is stored beginning at the next full table after those already in use). If a new WHERE clause is found it is stored via TABMNT in utility table 2. In the storing process its alphabetic instructions which referenced DEFINE names are converted as per step six, section 6.10. Also, whatever values are input for the CLASS and SIDE attributes are saved. When the WHERE clause is completely processed, it is moved from utility table 2 into the input instruction tables (see figure 110).

*Main routine of overlay link RPTPRN

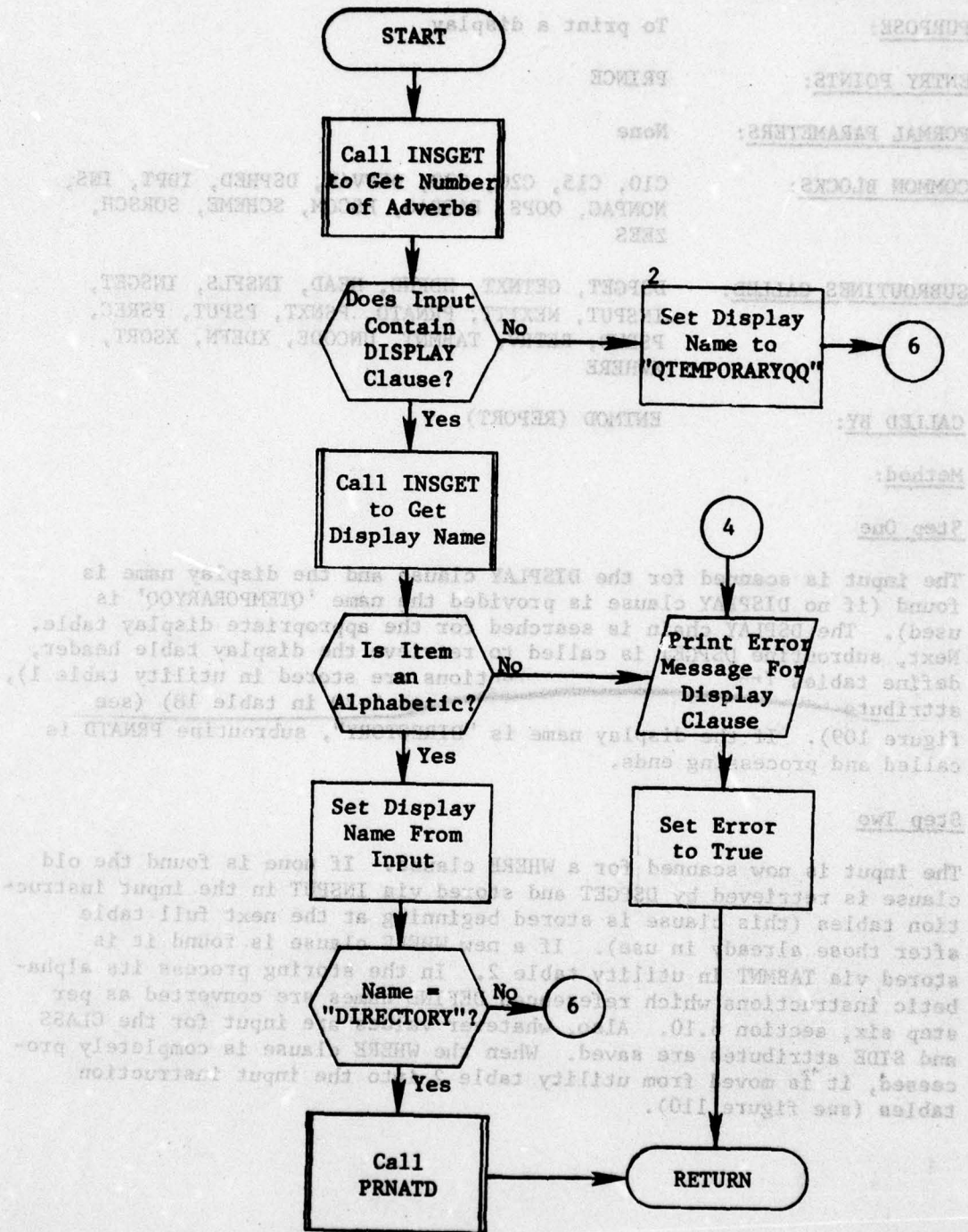


Figure 109. Subroutine PRINCE: Step One (Part 1 of 3)

6.11.2 Subroutine PRNATD

PURPOSE: To print the attribute directory

ENTRY POINTS: PRNATD

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C30

SUBROUTINES CALLED: HEAD, NEXTTT

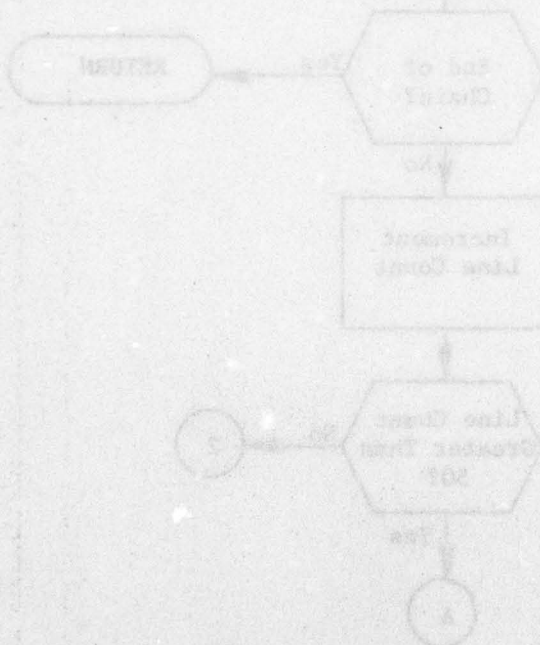
CALLED BY: PRINCE

Method:

After heading, the ATRIB chain is cycled. For each record retrieved, the attribute IATYPE is used to establish the mode (real, integer, alphabetic, and alphabetic with a list). For all but list alphabetics, a line is then printed displaying the pertinent information.

For list alphabetics, the list of values is retrieved from the VALIST chain and displayed along with the remainder of this information.

Subroutine PRNATD is illustrated in figure 114.1.



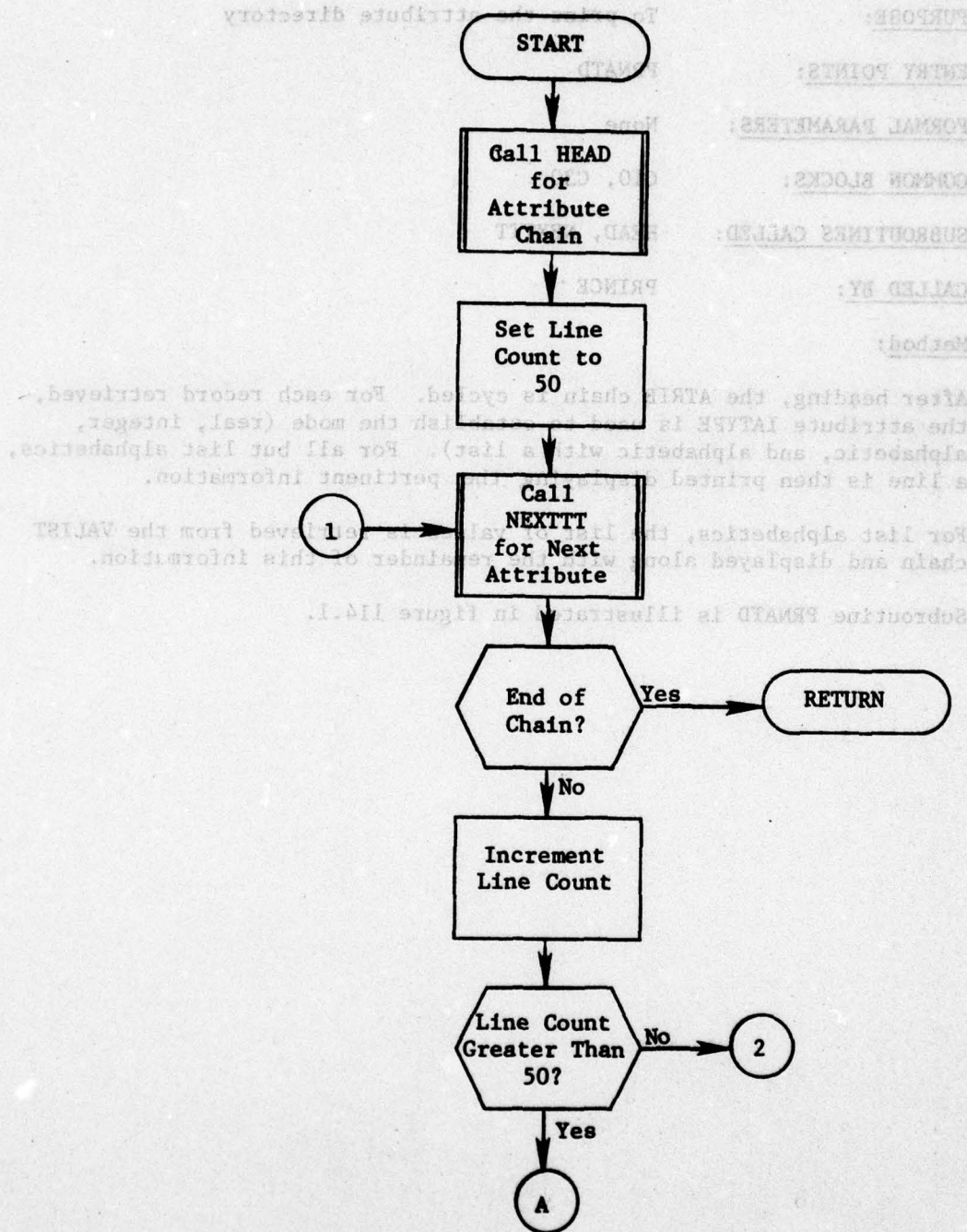


Figure 114.1. Subroutine PRNATD (Part 1 of 4)

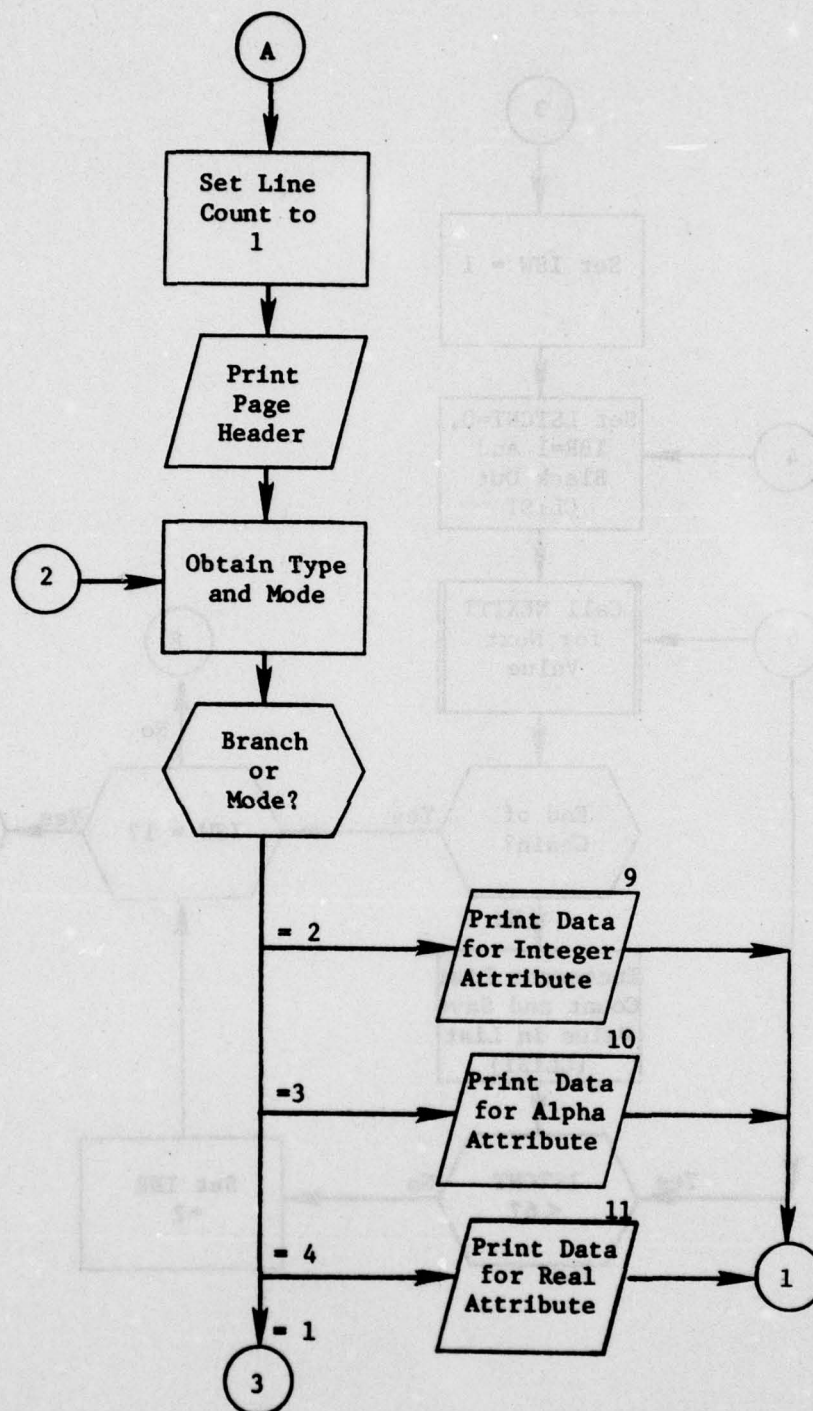


Figure 114.1. (Part 2 of 4)

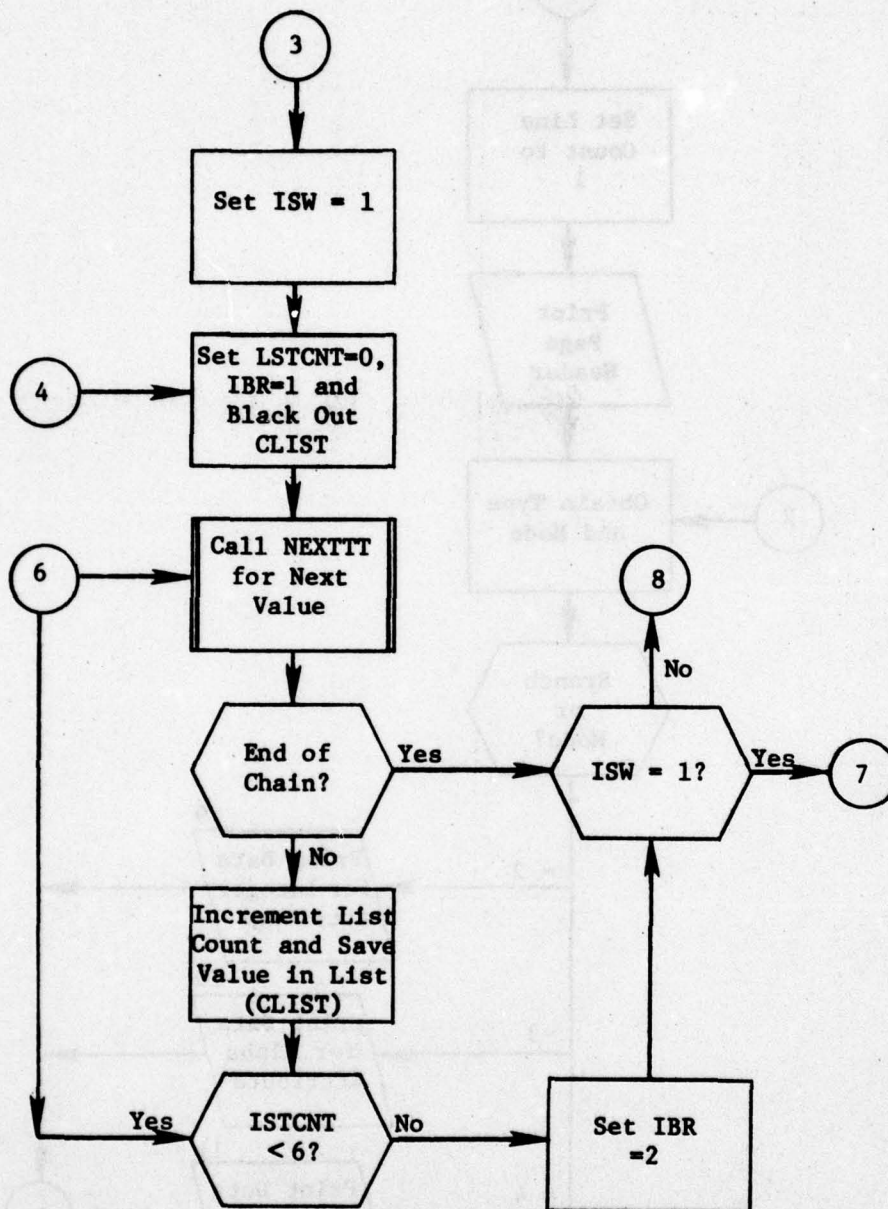


Figure 114.1. (Part 3 of 4)

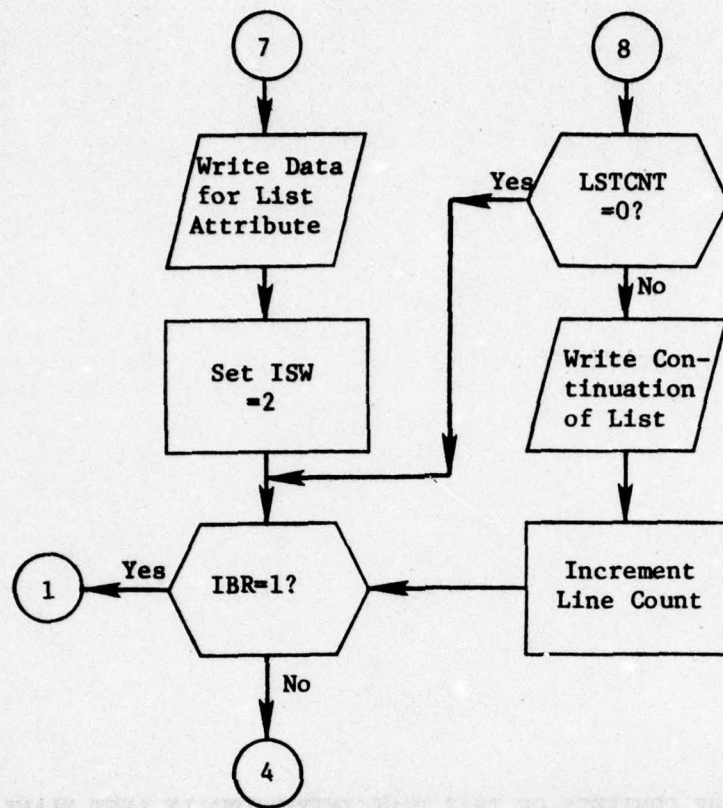


Figure 114.1. (Part 4 of 4)

SECTION 8. EXTERNAL INTERFACE MODULE (EIM)

8.1 Purpose

The purpose of the EIM is to create output tapes/files which are designed to be input to external processors.

8.2 Input

The output files to be built each have the precondition that all data necessary for the file be present in the data base.

8.3 Output

The output of EIM depends upon the verb and the FILE clause. BUILD FILE TABLE produces a single tape with six subsections. The format of this tape appears in table 21. BUILD FILE SIDAC produces two tapes, one containing BLUE targets, one containing RED targets. The format of the two tapes appears in table 22. BUILD FILE OTHER produces a single tape or file whose format and contents are specified by the user.

The PLOTIT and PLOTDATA verbs produce two tapes. One is a tape suitable for the CALCOMP plotter. The other contains information concerning all points which were not added to the plot tape owing to their being out of range of the plot size.

8.4 Concept of Operation

The EIM first determines which verb caused the call. If the verb was PLOTIT or PLOTDATA, the appropriate routine is executed. If the verb was BUILD, the FILE clause is found along with the special word within the clause. If the special word is SIDAC or TABLE the appropriate subroutines are called to produce the named files. If the special word is OTHER, the BLDOTH subroutine is called to produce the named files. If the special word is OTHER, the BLDOTH subroutine is called to produce the input defined file.

8.5 Identification of Subroutine Functions

8.5.1 Subroutine SIDAC. This subroutine produces the data base assessment tapes (DBASSESS). A preset retrieval scheme for all BLUE targets is set up and executed. Each target retrieved is written out in a modified JAD format. When all BLUE targets have been output, the scheme is modified for RED targets and the RED targets are written onto a separate tape.

Table 21. BUILD FILE TABLE Output File Formats
(Part 1 of 5)

TARGET LIST

<u>Column</u>	<u>Meaning</u>
1-8	'FITARGET'
9	Side: 1 for Blue; 2 for Red
10-14	Line Count, numeric
15	Blank
16-20	DESIG, alphabetic
21-24	Blank
25-31	Latitude (LAT), degrees, minutes, seconds
32-39	Longitude (LONG), degrees, minutes, seconds
40	Blank
41-46	NAME, alphabetic
47-50	World Area Code (WACNO), alphabetic
51-55	Bomber Encyclopedia Number (BENO), alphabetic
56	Blank
57-61	Category (CATCODE), numeric
62-63	Country Location (CNTRYL), alphabetic
64-69	Major Complex Number (MAJOR), numeric
70-71	TASK, alphabetic
72-76	Index Number (INDEXNO), numeric
77	Blank
78-80	Complex Number (ICOMPL), numeric

8.5.2 Subroutine TABLE. This subroutine produces the table file. The following tables are produced:

- o Target list
- o Vehicle characteristics list
- o Weapon characteristics list
- o Missile base list
- o Bomber base list
- o Offensive recovery base list

For each list a preset retrieval scheme is executed and the records retrieved are written onto the output tape.

8.5.3 Subroutine BLDOTH. This subroutine builds an output file according to user specified formats. This option is very similar in concepts to the REPORT module. First, all input clauses are scanned for attributes and a retrieval scheme is built. Next, all DEFINES are placed in proper execution order and DEFINE variable execution tables are built. Next, the WHERE clause is scanned for DEFINES and altered to handle them properly. The sort scheme is now created. Next, the retrieval scheme is executed and a file built of the resultant records and this file is sorted. Finally, the FORMAT clause is executed to produce the desired file (see figure 116).

8.5.4 Subroutine PLOTDATA. This subroutine builds an output plot tape. Four types of geographic data may be plotted:

- o Penetration corridors
- o Depenetration corridors
- o Refuel points
- o Recovery bases

For each desired set of data a preset retrieval scheme is used to retrieve the appropriate data. For each data record retrieved, the coordinates (LAT, LONG) are converted and/or scaled to the desired map characteristics and are processed for the plotter.

8.5.5 Subroutine PLOTIT. This subroutine builds an output plot tape. The plots produced are of bomber or tanker sorties. User selected sorties are retrieved and their events are plotted in sequence with special symbols used to distinguish events (see CSM UM 9-77, Volume I).

8.6 Common Blocks

The common blocks internal to EIM are listed in table 23.

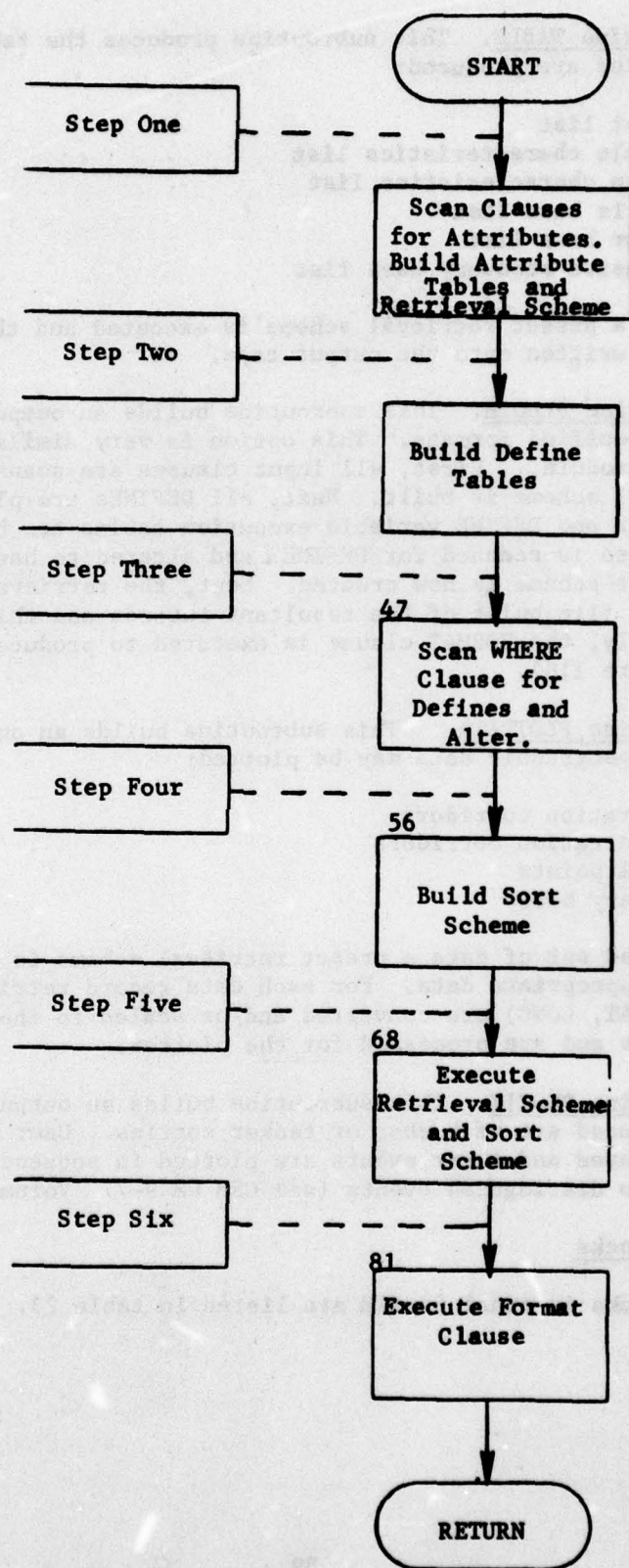


Figure 116. Subroutine BLDOTH: Macro Flow
590

Table 23. EIM Internal Common Blocks (Part 1 of 4)

<u>BLOCK</u>	<u>ARRAY OR VARIABLE</u>	<u>DESCRIPTION</u>
ATLST		Provides communication with the ATFNDR utility subroutine
	ATNUMB(100)	Attribute's identifying number
	ATADD(100)	Attribute's address
	ATTYP(100)	Attribute's mode (=1, integer, =2, alphabetic, =3, floating point)
	ATRA(100)	Attribute's lower limit
	ATRB(100)	Attribute's upper limit
	NUMAT	Number of attributes
DEFNMZ	DFNAME(100)	DEFINE variable name
	DFPNT(100)	Points to DEFINE clause in the Instruction code
DEFVAR	VARXX(100)	Value of DEFINE variable
DSPFRM		Provides communication with FORMAK utility subroutine
	FORMAR(50)	Format being constructed
	IFPNT	Pointer to next character
	IFLNG	Number of words used
EVENTS		Provides numeric identification for sortie events. Block is data preset
	LAUNM	Missile launch (set to 1)
	LAUNB	Bomber launch (set to 2)
	LEREFUEL	Refuel event (set to 4)
	LOCLATTR	Local attrition (set to 8)
	LAUNASM	Launch ASM (set to 14)
	LAUNDCOY	Launch decoy (set to 15)
	LANDHO	Land bomber (set to 16)
	LOHI	Change altitude (set to 17)
	MISSATTR	Missile attrition (set to 9)
	LEGDOG	Dogleg (set to 20)
	LABORT	Abort bomber (set to 13)
	LENTREF	Enter refuel area (set to 11)

Table 23. (Part 2 of 4)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
EVENTS (cont.)	LEAVEREF	Leave refuel area (set to 11)
	IGOHI	Go high (set to 18)
	IGOLO	Go low (set to 19)
	ITIME	Change time (set to 10)
	KSMTGT	ASM on target (set to 0)
INCRT	INCRT	Pen change time increment
PLNTAPE	KPL(90)	Sortie event place code
	JTP(90)	Sortie event identifier code
	HLA(90)	Sortie event latitude
	HLO(90)	Sortie event longitude
	IWH(90)	Sortie event warhead index
	CMT(90)	Sortie event cumulative time
	MHLOW(2)	Sortie go low events
	MHHI(2)	Sortie go high events
	MYGROUP	Sortie group number
	MYCORR	Sortie corridor number
	IOUTSRT	Sortie identification number
	LTOT	Number of sortie events
	LPLAN	Number of planned events
PLTARRYS	This block is used to store sortie data to be produced at one time	
	ILOW	Index to next available entry in arrays
	SUBHLA(202)	Event latitudes
	SUBHLO(202)	Event longitudes
	SUBJTP(200)	Event codes
	SUBKPL(200)	Event place codes
	IWARHD(200)	Event warhead indexes
	CTIME(200)	Cumulative time to event
	MYGROUP(10)	Sortie group numbers
	MYCORRI(10)	Sortie corridor numbers

Table 23. (Part 3 of 4)

<u>BLOCK</u>	<u>ARRAY OR VARIABLE</u>	<u>DESCRIPTION</u>
PLTARRAYS (cont.)	IOUTSORT(10)	Sortie identification number
	IBEG(10)	Beginning of sortie in event arrays
	IEND(10)	End of sortie in event arrays
	MHMINA(10)	First go low event in sortie
	MHMAXA(10)	First go high event in sortie
PLTPRO	MERCAT	Project type (set to 0)
	IDIREC	Indicator of plot direction (+1 for counter clockwise, -1 for clockwise)
	FLMDAO	Longitude of origin
	PHIO	Latitude of origin
	THETAO	Angle between meridian and X-axis
	PHI1	Standard parallel closest to equator
	PHI2	Standard parallel closest to pole
	A1L10	Fractional part of log for PHI1
PLTSPE	A2L10	Fractional part of log for PHI2
	ISIZE	Plot size indicator =0 for 50 x 40 =1 for 20 x 20 =2 for 10 x 10
	XAXLEN	Length of X-axis in inches
	YAXLEN	Length of Y-axis in inches
	FACTOR	Number of plots per page
	SCALE	Ratio of world units to plot units
	PRINSP	PRINON Switch to control optional print =True, produce print =False, do not produce print
PSCOM		Used to communicate with PSREC utility
	RECORD(100)	Body of print/sort record
	RECLN	Number of words in record

Table 23. (Part 4 of 4)

<u>BLOCK</u>	<u>ARRAY OR VARIABLE</u>	<u>DESCRIPTION</u>
RTLST	RTLIST(100)	Used to communicate with ATFNDR utility List of record type numbers for retrieval
	NUMREC	Number of record types in list
	HDREC	Record type name of primary header
	HCLASS	CLASS value of primary header
	HSIDE	SIDE value of primary header
	HOPT	CLASS/SIDE option for scheme
	JHDR	Record type number of primary header
SCHEME	POINT	Pointer to current instruction of retrieval scheme
	SCHEME(200)	Retrieval scheme (see UM 9-77, Vol. I, section 5.3.2.4)
SLIST	NLIST	Number of sorties in list
	LIST(600)	List of desired sortie numbers
SORSCH	SRTSCH(100)	Sort scheme (see section 6.10)
TAPES	PLOTTA	Logical unit number for plot tape
	PIECTA	Logical unit number for tape for nonplotted points
XMEDGE	XMEDGE	Map edge
	XLL	X-coordinate of last point
	YLL	Y-coordinate of last point
	XL	X-coordinate of point to be plotted
	YL	Y-coordinate of point to be plotted
	KWEDGE	Converted value for latitude of origin
	BANGL	Converted value for longitude of origin
ZEES	Used internally for core position only	
	ZA	Equivalent to output array from INSGET
	ZB	
	ZC	
	ZD	
	ZE	

8.7 Subroutine ENTMOD

PURPOSE: Entry subroutine for EIM

ENTRY POINTS: ENTMOD (first subroutine called when overlay EIM is executed)

FORMAL PARAMETERS: None

COMMON BLOCKS: OOPS, PRINSP

SUBROUTINES CALLED: BLDOTH, INSGET, PLOTDA, SIDAC, TABBLE

CALIED BY: MODGET

Method:

First the input is scanned for the ONPRINTS adverb and if found the PRINON switch is set to true. Next, the verb is retrieved, if it is PLOTDATA the overlay link and PLOTDA are executed. If the verb is PLOTIT the overlay link for PLOTIT is executed. If the verb is BUILD, the FILE clause is found and the special word checked. If it is TABLE, the overlay for TABBLE is called. If it is SIDAC, the overlay link for SIDAC is called. If it is OTHER the overlay link for BLDOTH is called.

Subroutine ENTMOD (EIM) is illustrated in figure 117.

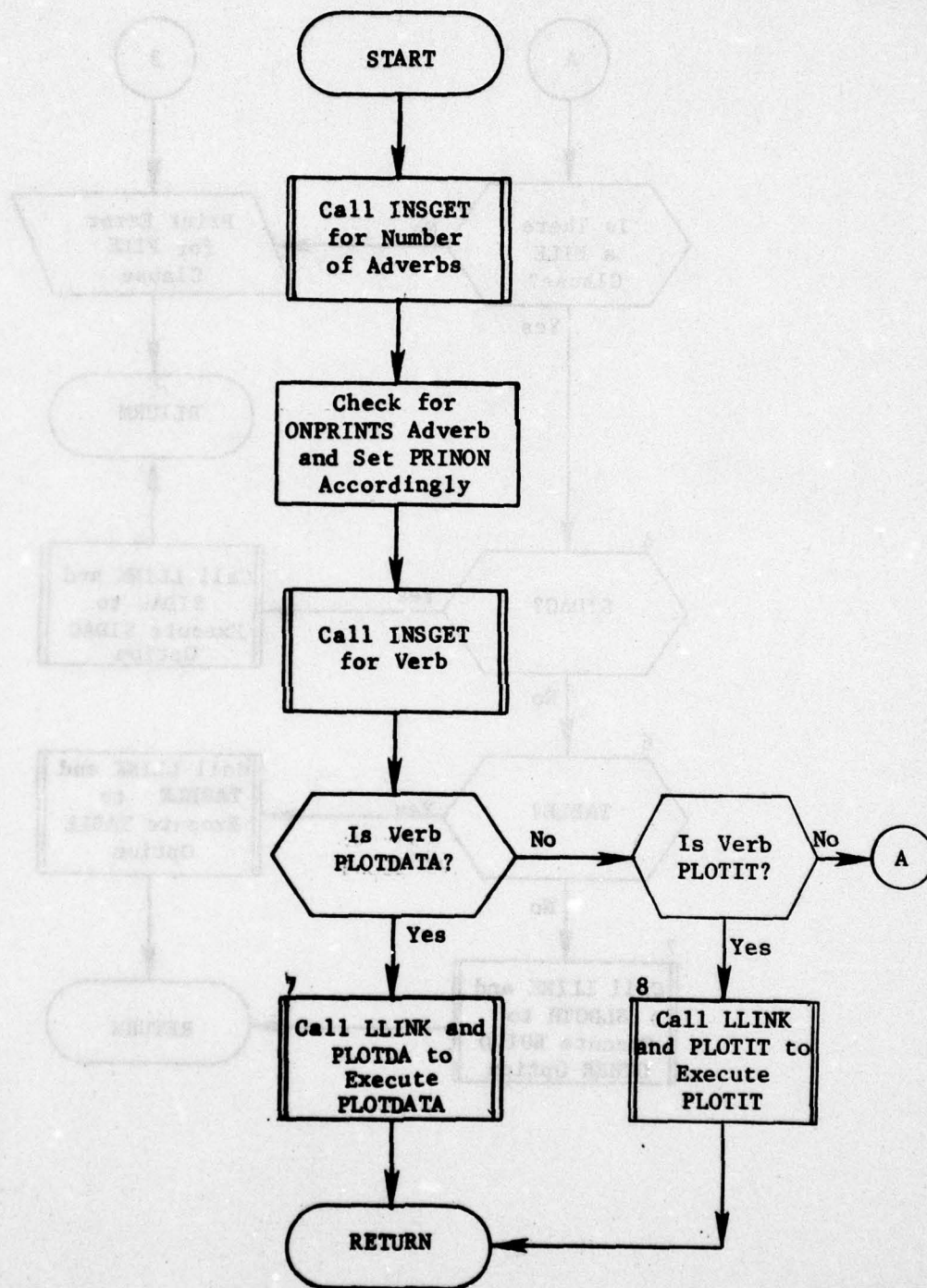


Figure 117. Subroutine ENTMOD (EIM) (Part 1 of 2)

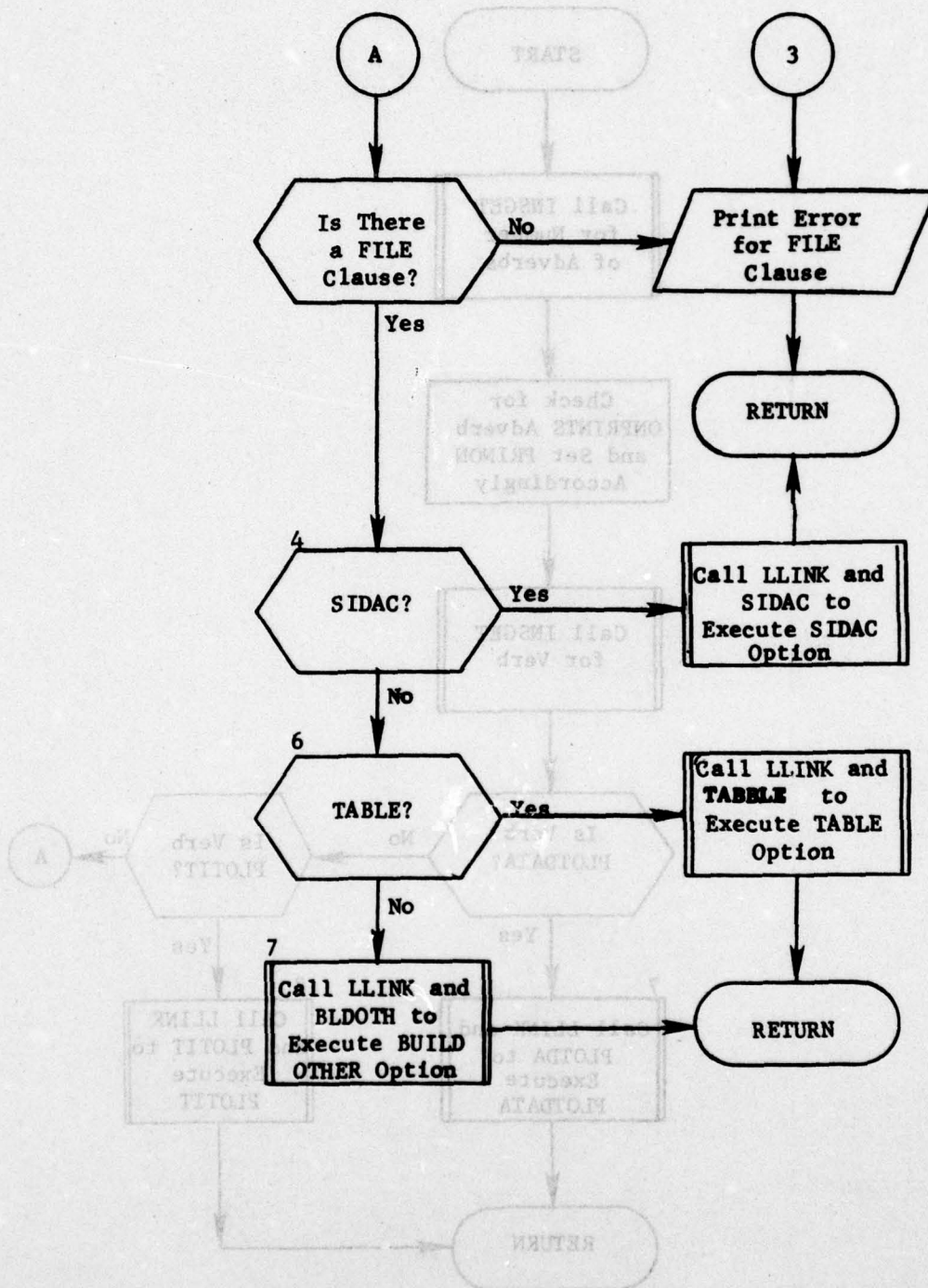


Figure 117. (Part 2 of 2)

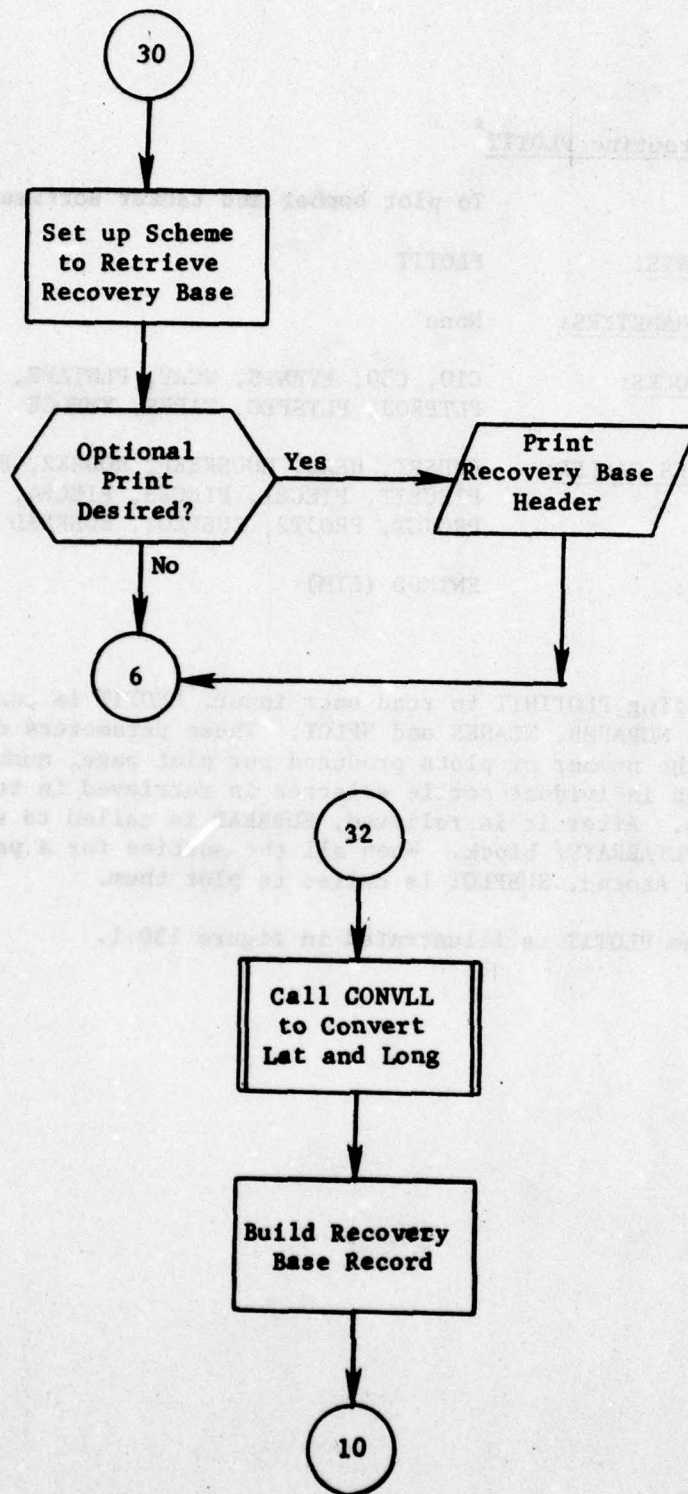


Figure 130. (Part 13 of 13)

8.12 Subroutine PLOTIT*

PURPOSE: To plot bomber and tanker sorties

ENTRY POINTS: PLOTIT

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C30, EVENTS, WCRT, PLNTAPE, PLTARRYS, PLTPROJ, PLTSPEC, TAPES, XMEDGE

SUBROUTINES CALLED: FNDSTR, HEAD, HOUSKEEP, HOUSK2, HOUSK3, NEXTTT, PIECEIT, PIECE1, PIECE3, PIECE4, PLOTINIT, PROJCT, PROJ2, SUBPLOT, SUBREAD

CALLED BY: ENTMOD (EIM)

Method:

After calling PLOTINIT to read user input, PLOTIT is controlled by the variables NGRAPHs, NCASES and NPLOT. These parameters drive loops that control the number of plots produced per plot page, number of pages, etc. Each individual sortie selected is retrieved in turn in the inner most loop. After it is retrieved, SUBREAD is called to store the sortie in the /PLTARRAYS/ block. When all the sorties for a particular graph have been stored, SUBPLOT is called to plot them.

Subroutine PLOTIT is illustrated in figure 130.1.

* Main routine of overlay PLOTIT.

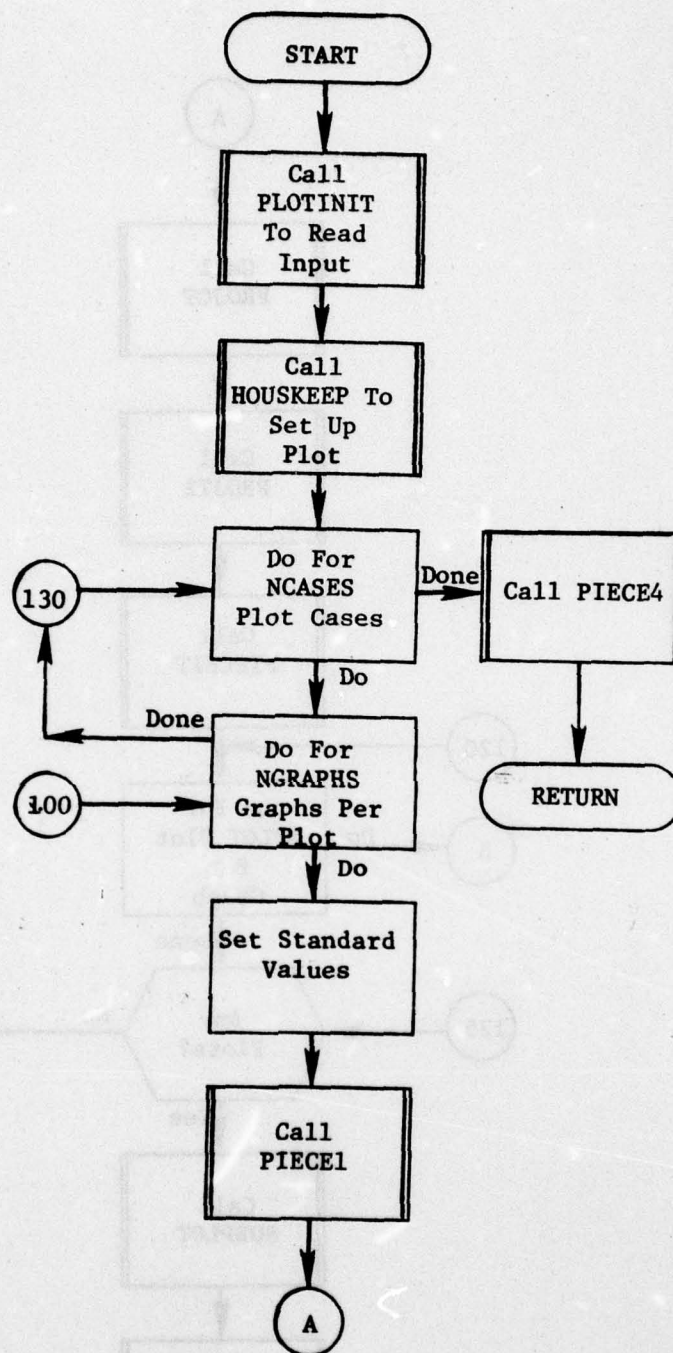


Figure 130.1. Subroutine PLOTIT (Part 1 of 4)

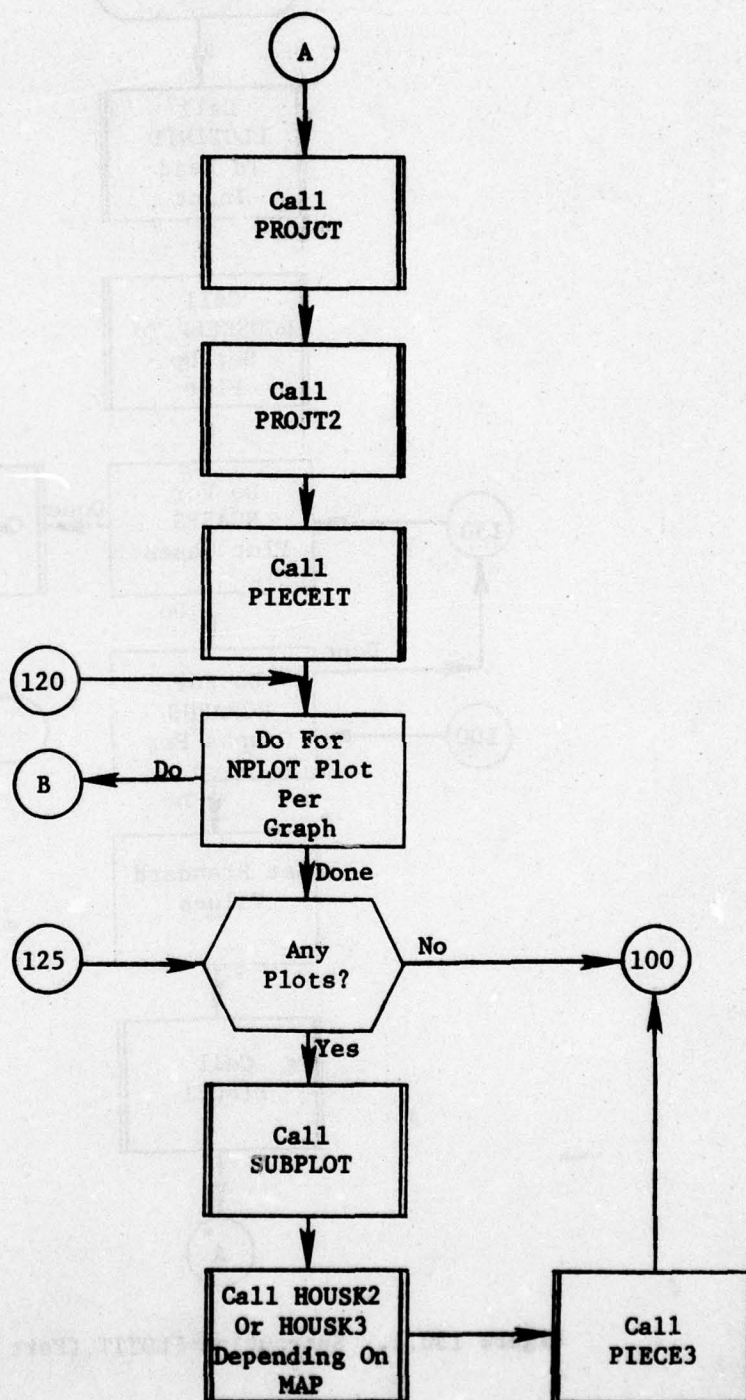


Figure 130.1. (Part 2 of 4)

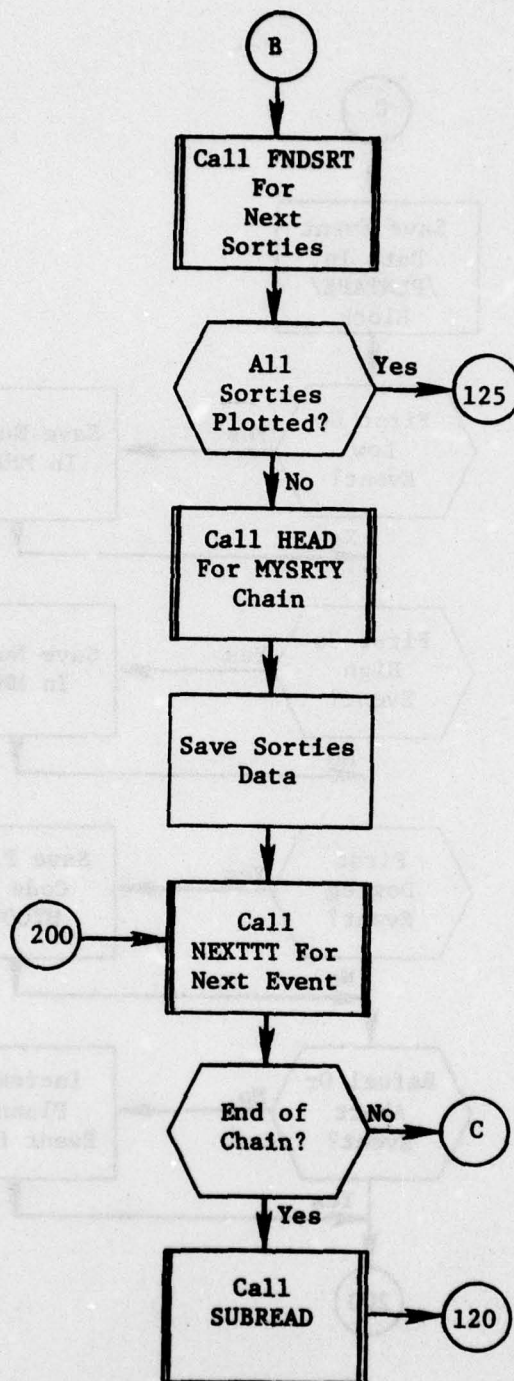


Figure 130.1. (Part 3 of 4)

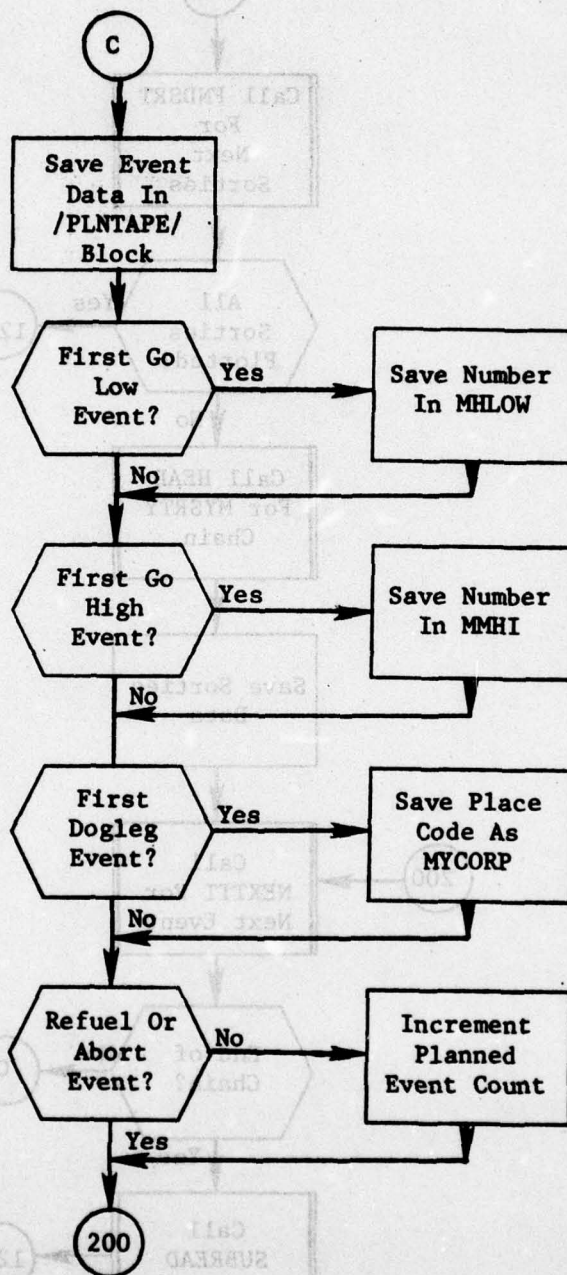


Figure 130.1. (Part 4 of 4)

8.12.1 Subroutine FNDSTRT

PURPOSE: To retrieve the next desired sortie

ENTRY POINT: FNDSTRT

FORMAL PARAMETERS: ISWIT: 1 = Initialize search
2 = Get next sortie in normal sequence
3 = Get next sortie in /SLIST/ sequence
4 = All sorties retrieved

COMMON BLOCKS: C10, C15, C30, SLIST

SUBROUTINES CALLED: ABORT, DIRECT, HDFND, NEXTTT, RETRV

CALLED BY: PLOTIT, PLOTINIT

Method:

The function depends upon value of ISWIT. If ISWIT is one, the sortie count is set to zero, and the sortie header is retrieved. Also, if NLIST is nonzero, the reference code array, BREAK is zeroed out.

If ISWIT is 2, the next sortie on the SORTIE chain is retrieved. If the end of the chain is encountered, ISWIT is set to 4.

If ISWIT is 3, the sortie count is incremented. The value in BREAK is checked. If nonzero, the reference code is used to retrieve the sortie. If zero, the SORTIE chain is cycled until the desired sortie is found. If any other desired sorties are encountered in this process their reference codes are saved. When all sorties have been retrieved ISWIT is set to 4.

If ISWIT is 4, the subroutine simply returns.

Subroutine FNDSTRT is illustrated in figure 130.2.



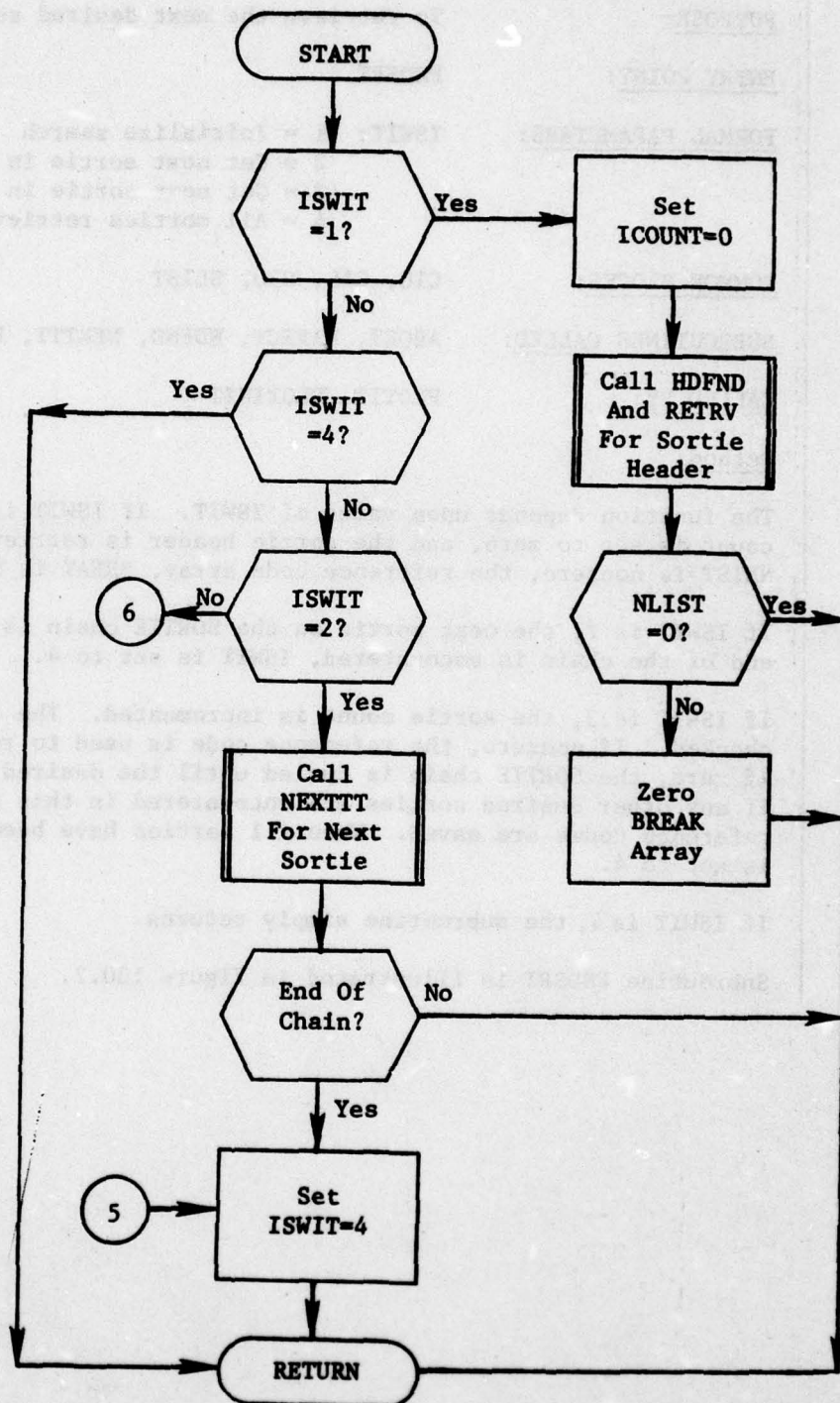


Figure 130.2. Subroutine FNDSTRT (Part 1 of 2)

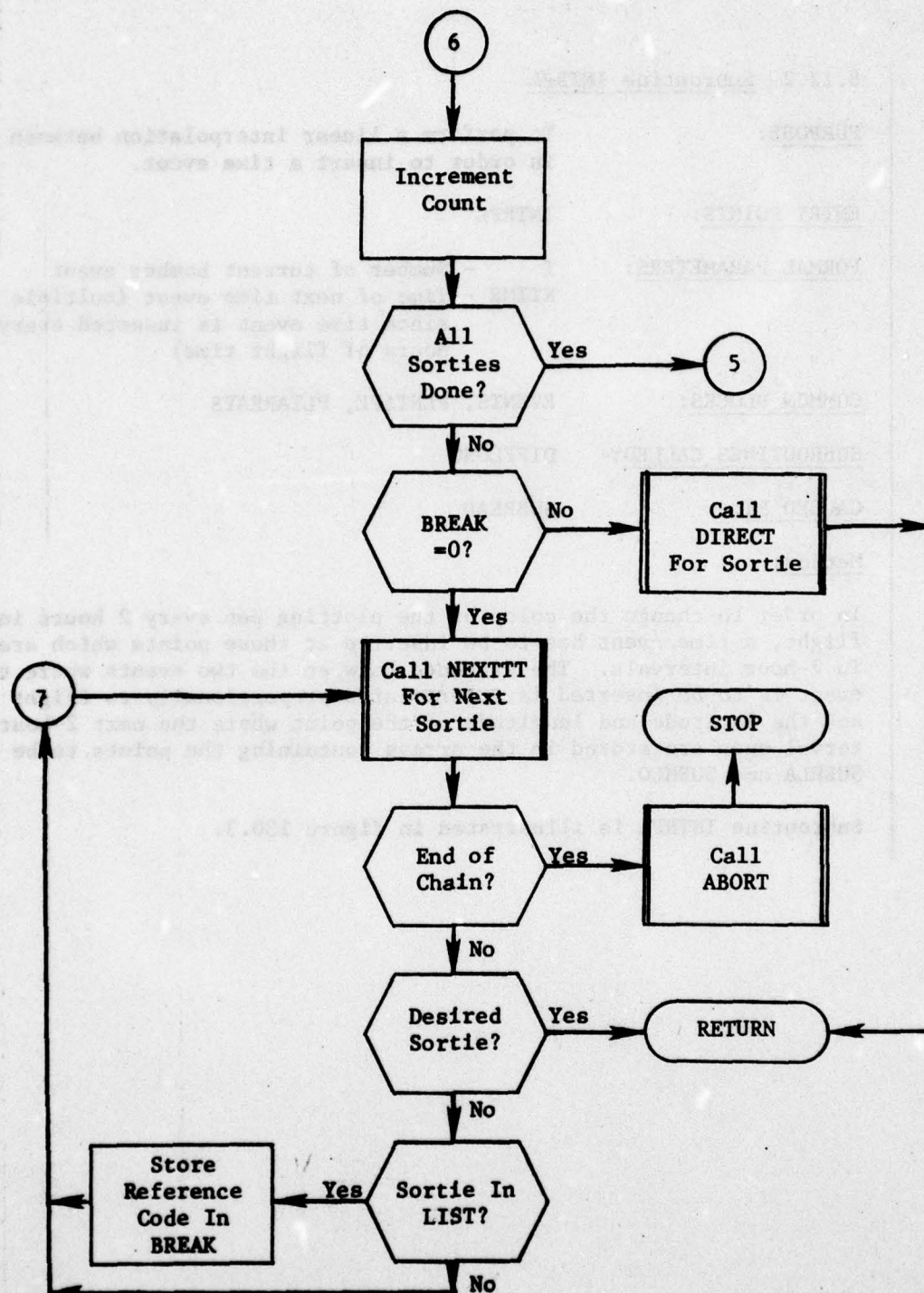


Figure 130.2. (Part 2 of 2)

8.12.2 Subroutine INTRPL

PURPOSE: To perform a linear interpolation between events in order to insert a time event.

ENTRY POINTS: INTRPL

FORMAL PARAMETERS:
 I - Number of current bomber event
 KTIME - Time of next time event (multiple of 2, since time event is inserted every 2 hours of flight time)

COMMON BLOCKS: EVENTS, PLNTAPE, PLTARRAYS

SUBROUTINES CALLED: DIFFLONG

CALLED BY: SUBREAD

Method:

In order to change the color of the plotting pen every 2 hours into the flight, a time event has to be inserted at those points which are reached in 2-hour intervals. The distance between the two events where the time event is to be inserted is interpolated proportionally to flight time and the latitude and longitude of the point where the next 2-hour interval ends are stored in the arrays containing the points to be plotted, SUBHLA and SUBHLO.

Subroutine INTRPL is illustrated in figure 130.3.

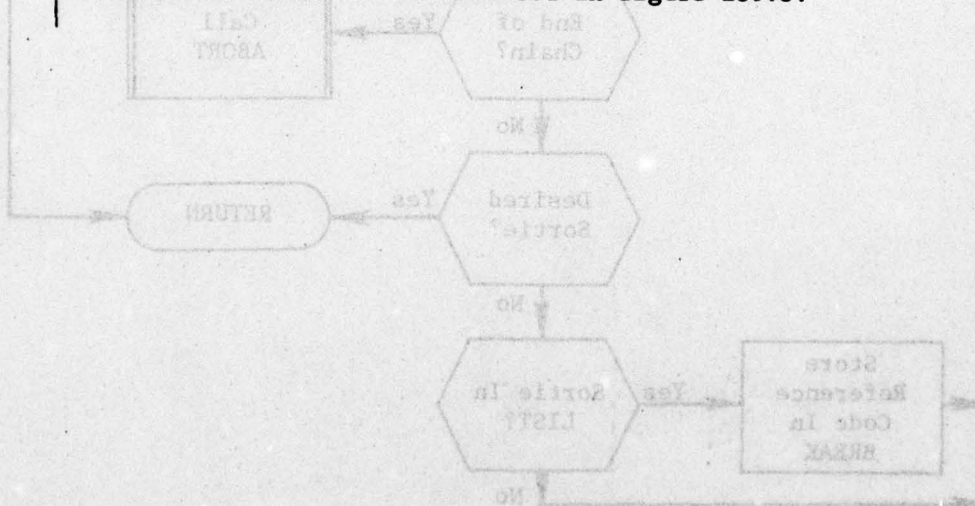


Figure 130.3 (Part 1 of 2)

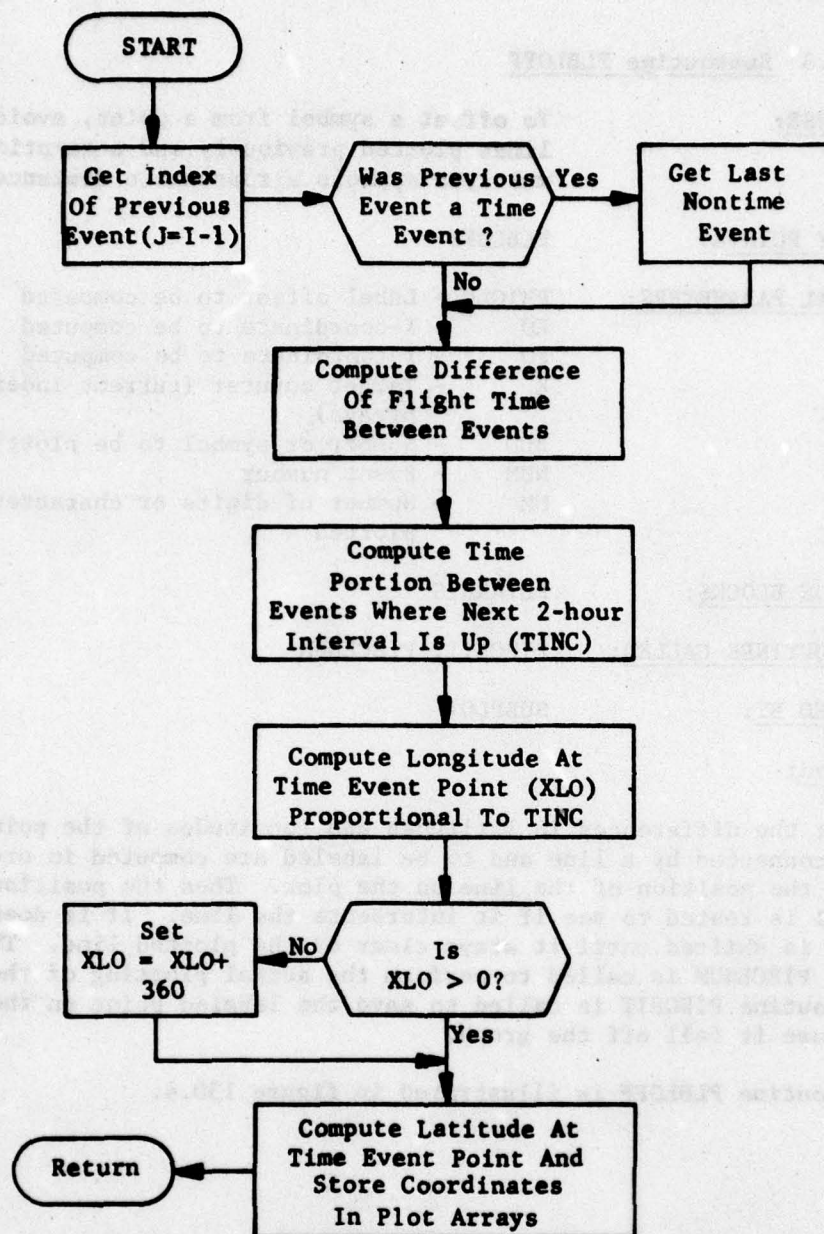


Figure 130.3. Subroutine INTRPL

8.12.3 Subroutine PLBLOFF

PURPOSE:

To offset a symbol from a point, avoiding the lines plotted previously and attempting to keep multiple symbols a reasonable distance apart.

ENTRY POINTS:

PLBLOFF

FORMAL PARAMETERS:

PHIOLD - Label offset to be computed
XO - X-coordinate to be computed
YO - Y-coordinate to be computed
K - Target counter (current index into data arrays)
BCD - Number or symbol to be plotted
NUM - Event number
MM - Number of digits or characters to be plotted

COMMON BLOCKS:

PLTARRYS

SUBROUTINES CALLED:

PIECEIT, PIECENUM

CALLED BY:

SUBPLOT

Method:

First the differences in latitudes and longitudes of the points which are connected by a line and to be labeled are computed in order to determine the position of the line on the plot. Then the position of the label is tested to see if it intersects the line. If it does, its position is shifted until it stays clear of the plotted line. Then subroutine PIECENUM is called to perform the actual plotting of the label. Subroutine PIECEIT is called to save the labeled point on the PIECTAPE in case it fell off the graph.

Subroutine PLBLOFF is illustrated in figure 130.4.

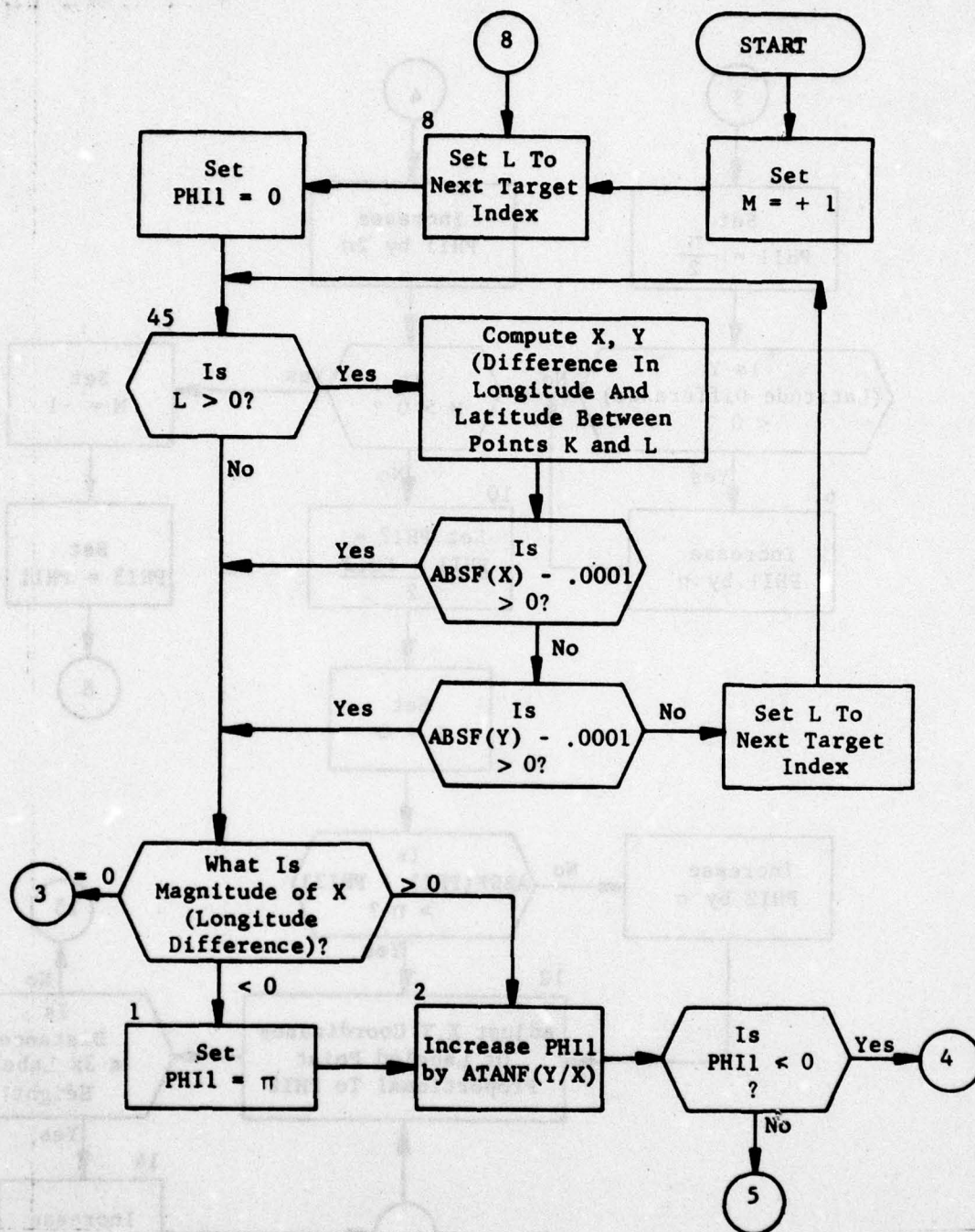


Figure 130.4. Subroutine PLBLOFF (Part 1 of 3)

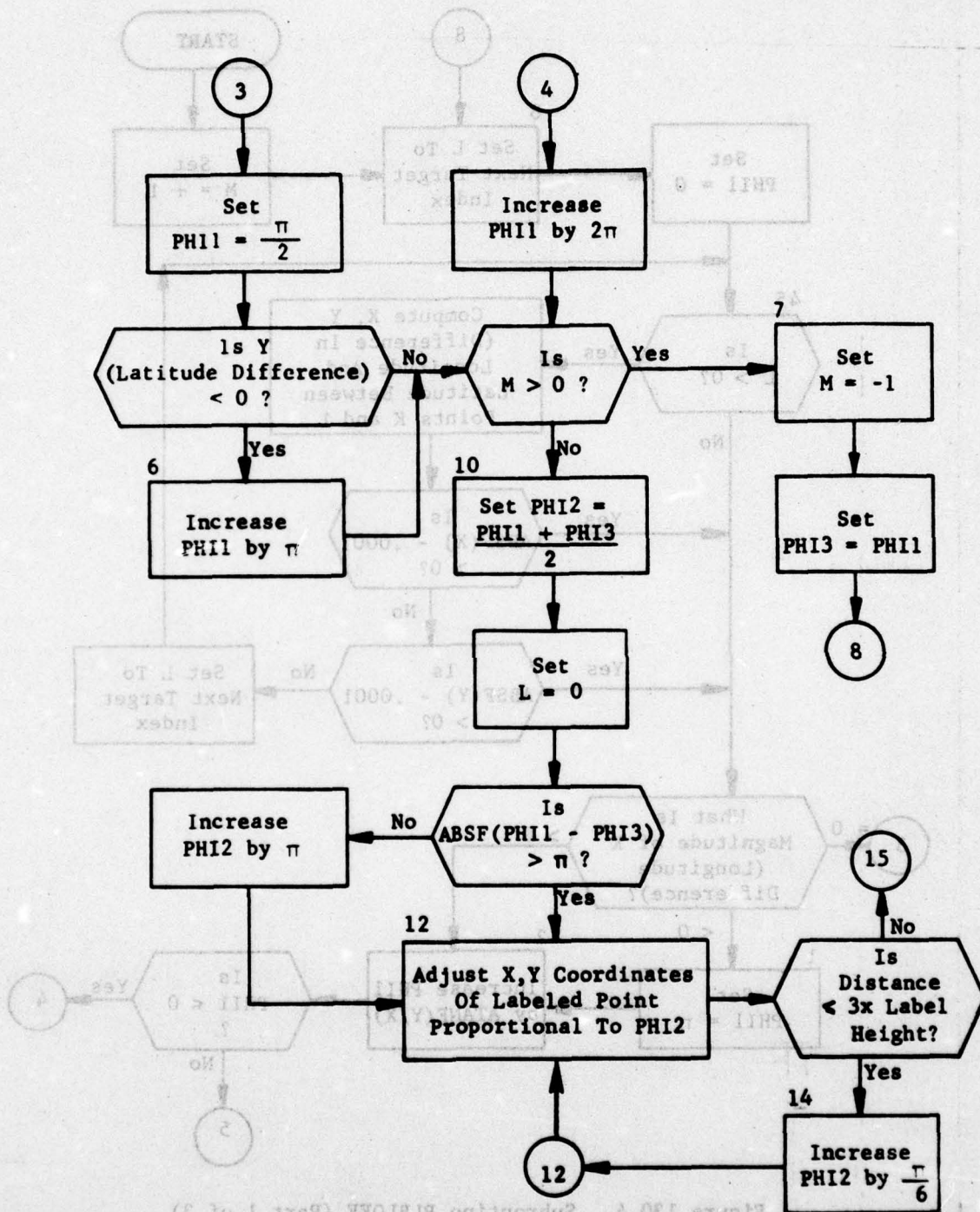


Figure 130.4. (Part 2 of 3)

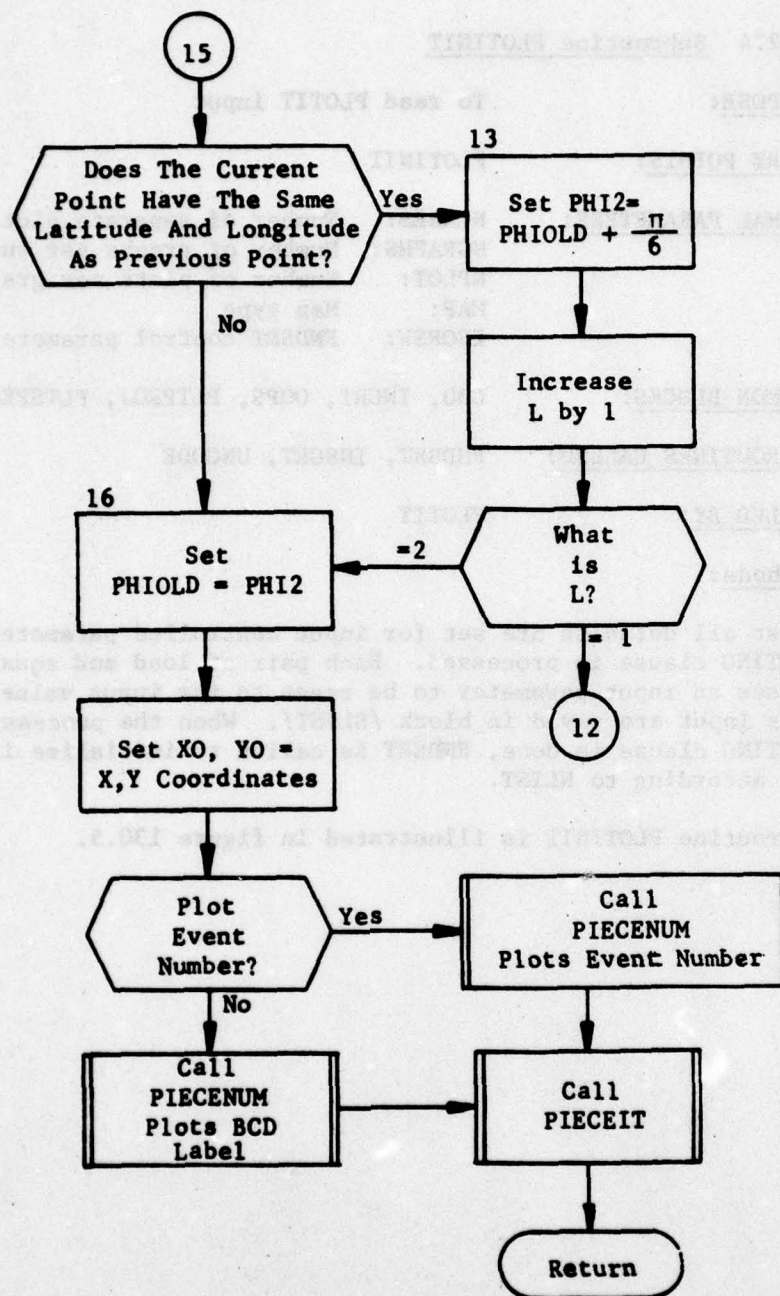


Figure 130.4. (Part 3 of 3)

8.12.4 Subroutine PLOTINIT

PURPOSE: To read PLOTIT input

ENTRY POINTS: PLOTINIT

FORMAL PARAMETERS:

NCASES:	Number of separate plot runs
NGRAPHS:	Number of graphs per run
NPLOT:	Number of plots per graph
MAP:	Map type
ISORSW:	FNDSTRT control parameter

COMMON BLOCKS: C30, INCRT, OOPS, PLTPROJ, PLTSPEC, SLIST, ZEES

SUBROUTINES CALLED: FNDSTRT, INSGT, UNCODE

CALLED BY: PLOTIT

Methods:

First all defaults are set for input controlled parameters. Then the SETTING clause is processed. Each pair of load and equals instruction causes an input parameter to be reset to the input value. Sortie numbers input are saved in block /SLIST/. When the processing of the SETTING clause is done, FNDSTRT is called to initialize it and ISORSW set according to NLIST.

Subroutine PLOTINIT is illustrated in figure 130.5.

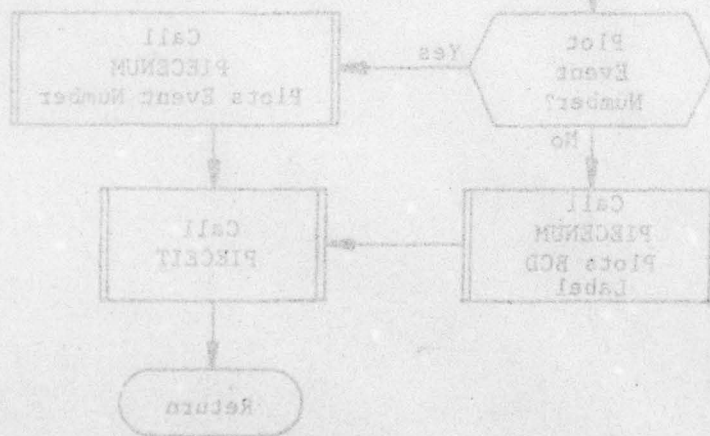


Figure 130.5 (Part 3 of 3)

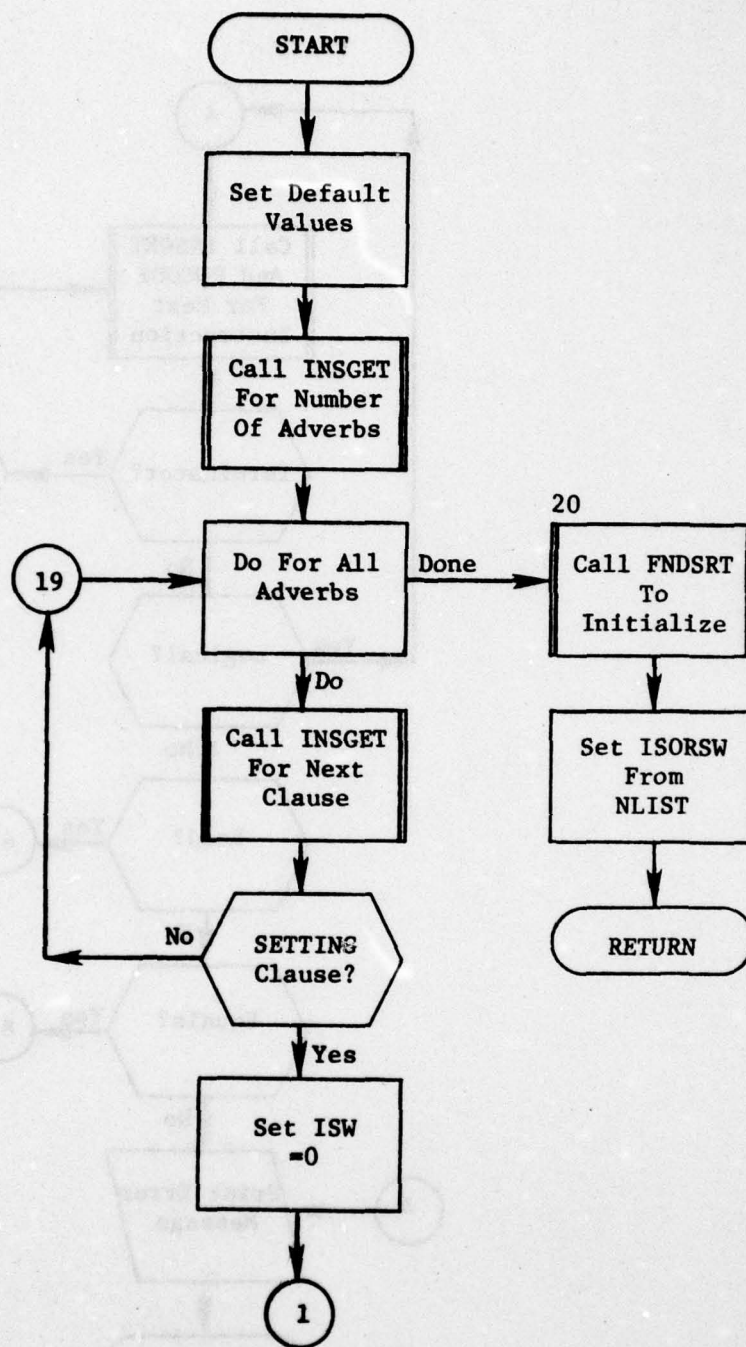


Figure 130.5. Subroutine PLOTINIT (Part 1 of 4)

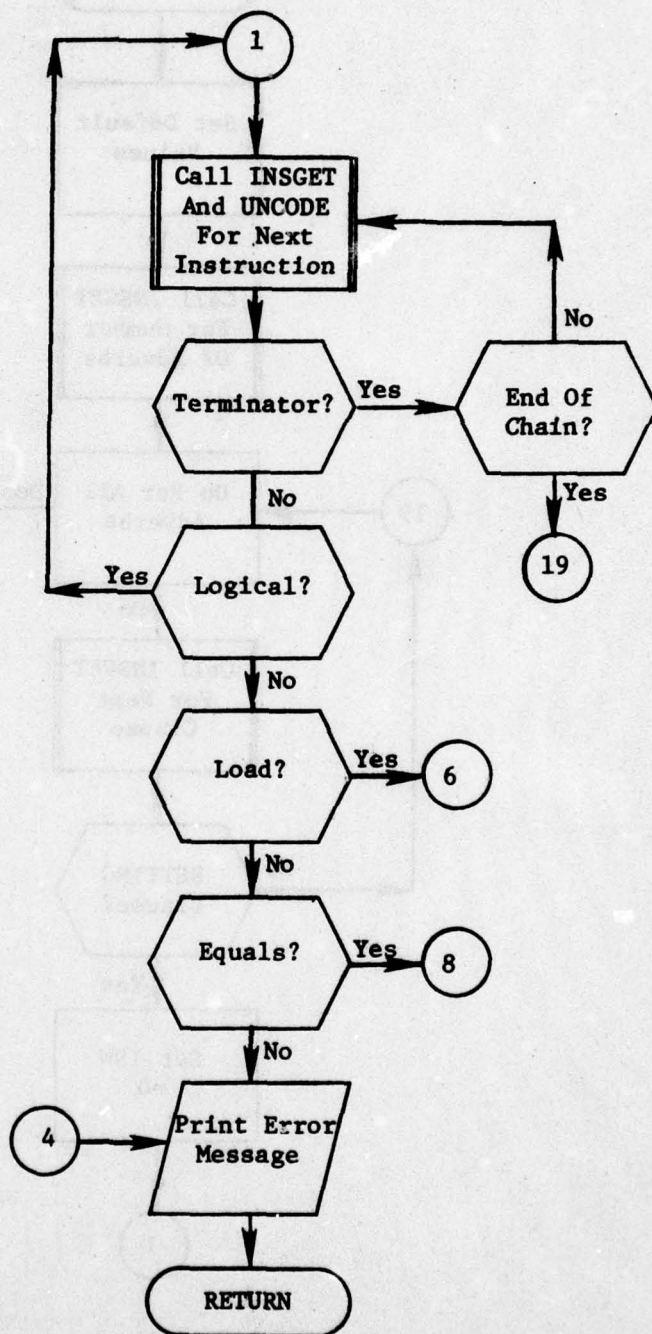


Figure 130.5. (Part 2 of 4)

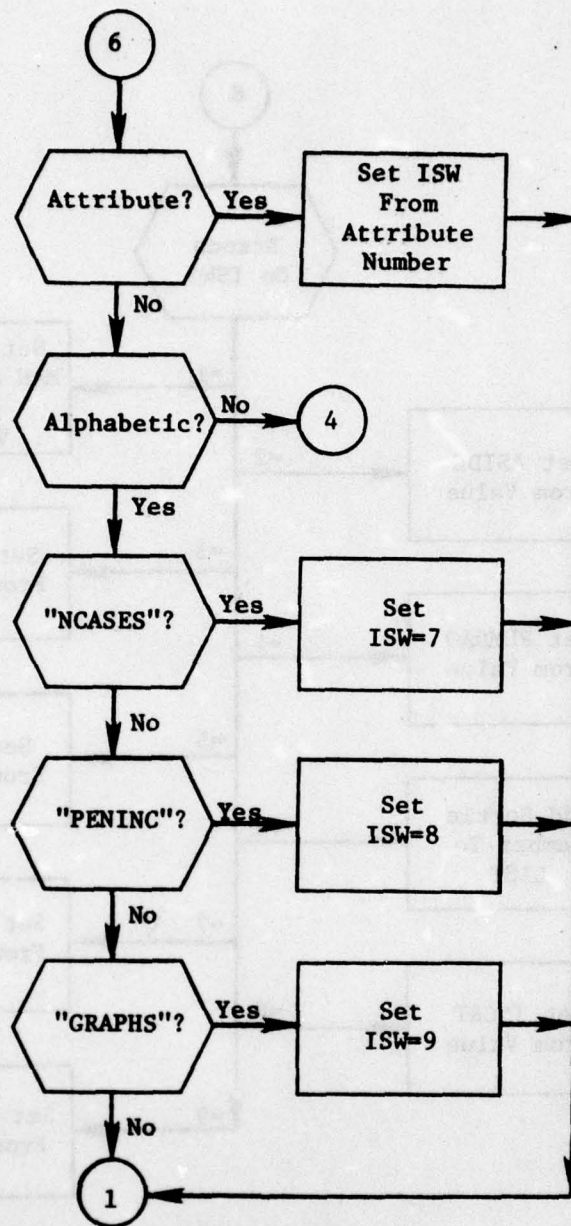


Figure 130.5. (Part 3 of 4)

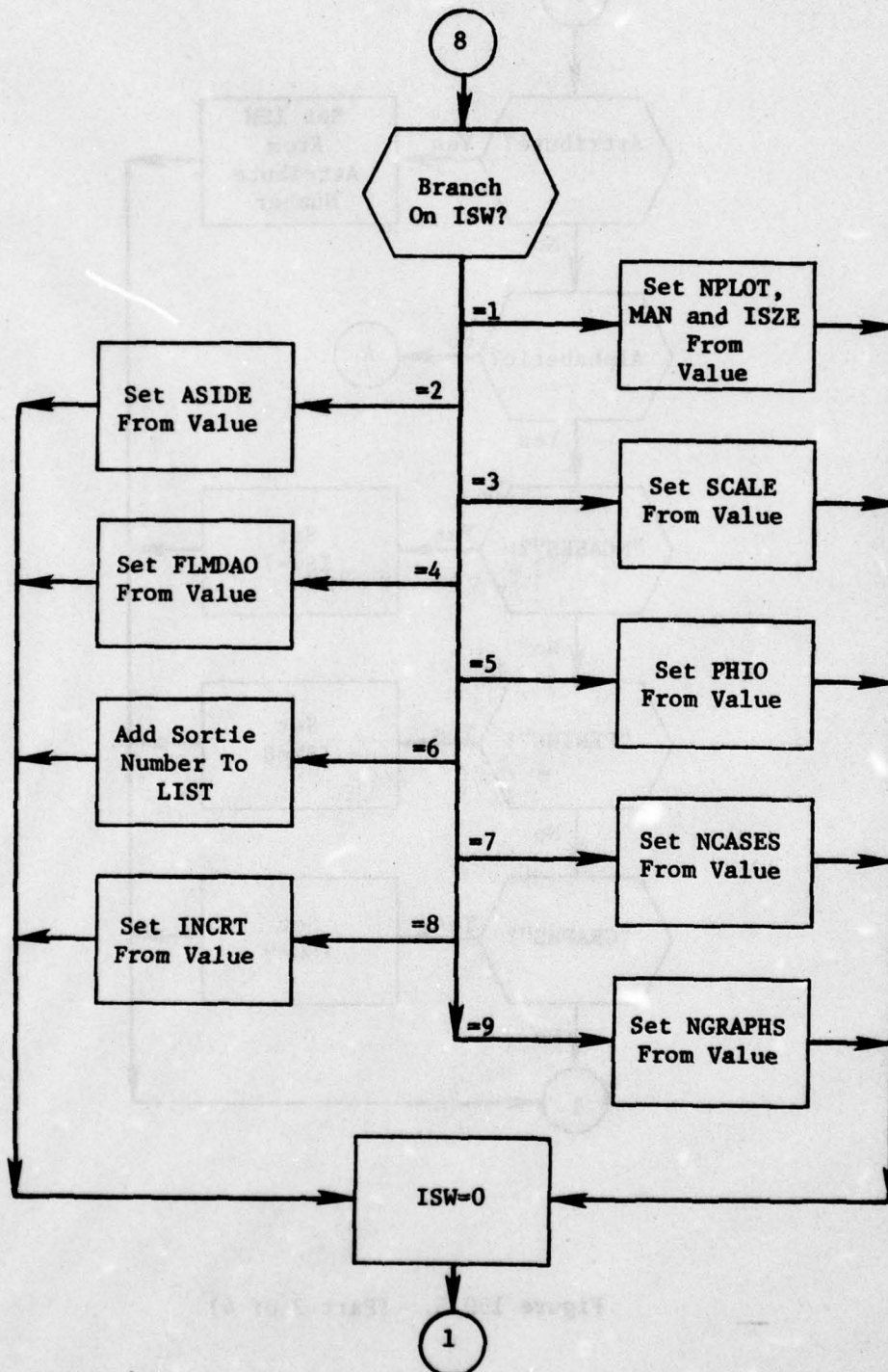


Figure 130.5. (Part 4 of 4)

8.12.5 Subroutine SUBPLOT

PURPOSE: To perform the actual plotting of the event data.

ENTRY POINTS: SUBPLOT

FORMAL PARAMETERS: NUMPLOT - Plot number
MAP - Map type indicator

COMMON BLOCKS: EVENTS, INMAP, PLTARRYS, PLTPROJ, PLTSPEC, TICIT, XMEDGE

SUBROUTINES CALLED: MAPEDGE, NEWPEN, * NUMBER, * PIECEIT, PIECENUM, PIKS, PLBOFF, PLOT, * PLTTIC, PRJCT4, SYMBOL*

CALLED BY: PLOTIT

Method:

First the map type indicator is tested to determine which projection subroutine has been chosen. If the point coordinates are to be plotted in the Mercator or Lambert projection, subroutine PROJCT3 is called to initialize the projection parameters. For rectilinear projection, the scaling parameters are computed and printed; then the arrays of latitudes and longitudes are scanned for their minimum and maximum values in order to find the best origin of the plot. Then the ticmarks, y-axis and x-axis are drawn and the labels are plotted. If subroutine PIKS is chosen to perform the point conversion, it is called to scale the latitudes and longitudes into inches along the plot axes.

Now that the origin of the plot is determined, the event points are converted to plotting units and the actual plotting of the events can begin. The following steps are repeated for all graphs: subroutine PIECEIT is called to initialize the pen for this plot. The first and last index into the data arrays for this plot are set. The coordinates of the first point are stored in XL, YL. Now all the events to be put on this plot are processed, repeating the following steps until the last index into the data arrays is reached: the coordinates of the previous point are stored in XLL, YLL, and XL, YL are set for the current point. Subroutines PIECEIT and MAPEDGE are called to put the point on the graph. Now the event type is tested for proper labeling of the point. The event type indicator, as stored in array SUBJTP, is compared to all possible event types: time, refuel, bomber launch, recovery, boundary crossing, to-to-low-altitude, go-to-high-altitude, launch decoy, launch ASM, local attrition or drop bomb, abort, enter or leave refuel area; when a match is found, subroutine PIECENUM is called to perform the proper labeling of the event.

* CALCOMP routines.

When the plotting of the event points is completed, the identifying parameters for the sorties, such as group number, corridor number and tanker or bomber sortie number are retrieved for all the events on the graph and printed on-line. The plotting routines NUMBER and SYMBOL are called to put all the identifying information on the plot.

Subroutine USBPLOT is illustrated in figure 130.6.

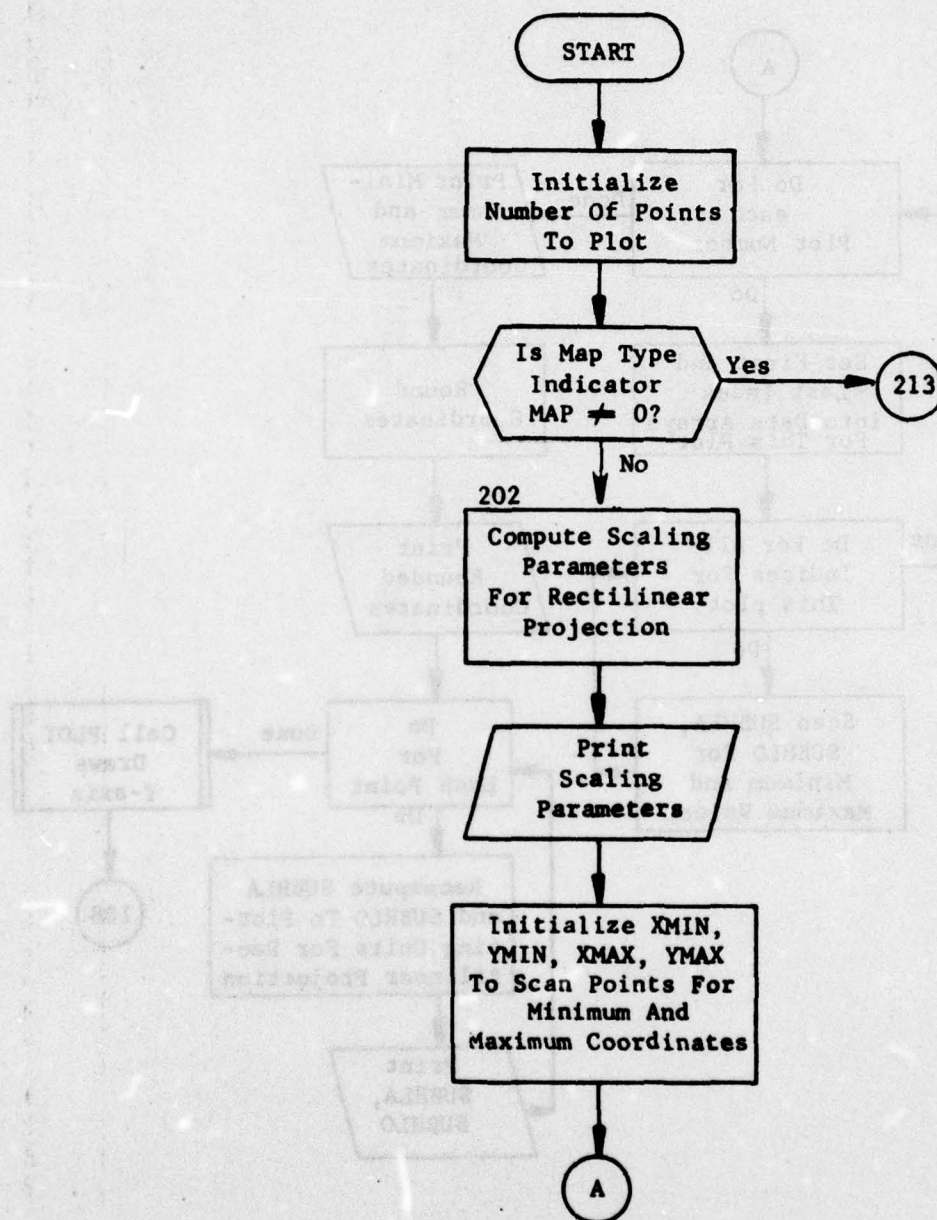


Figure 130.6. Subroutine USBPLOT (Part 1 of 13)

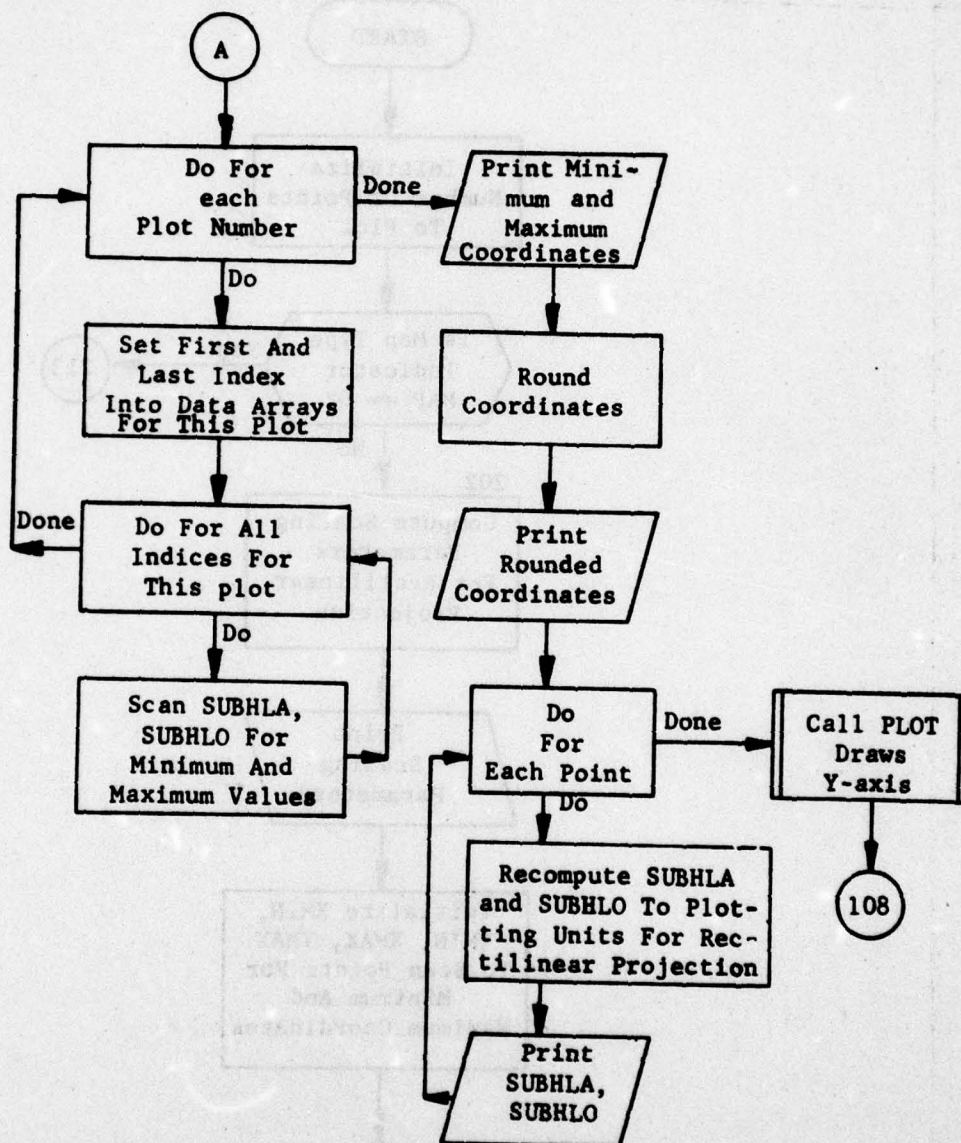


Figure 130.6. (Part 2 of 13)

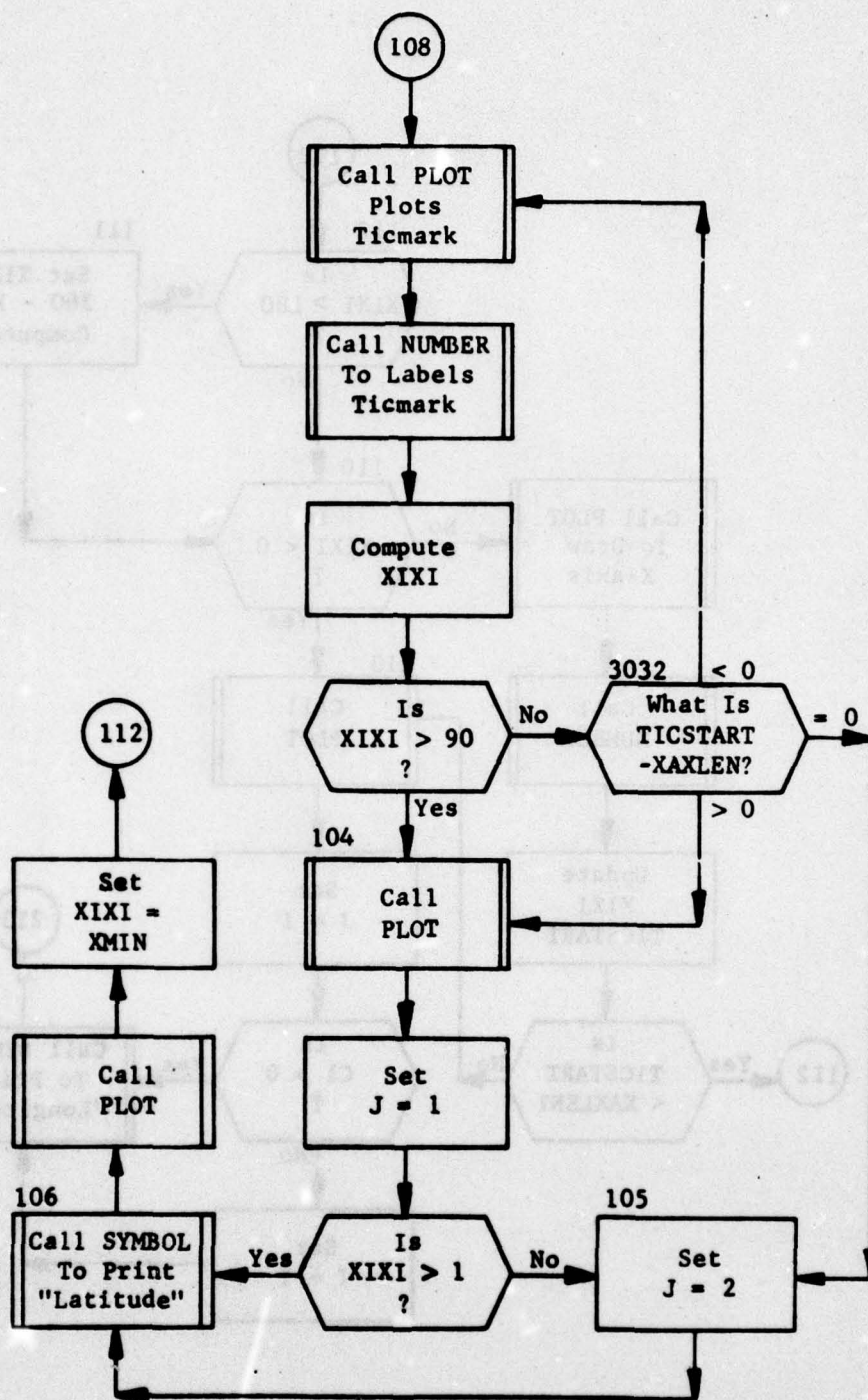


Figure 130.6. (Part 3 of 13)

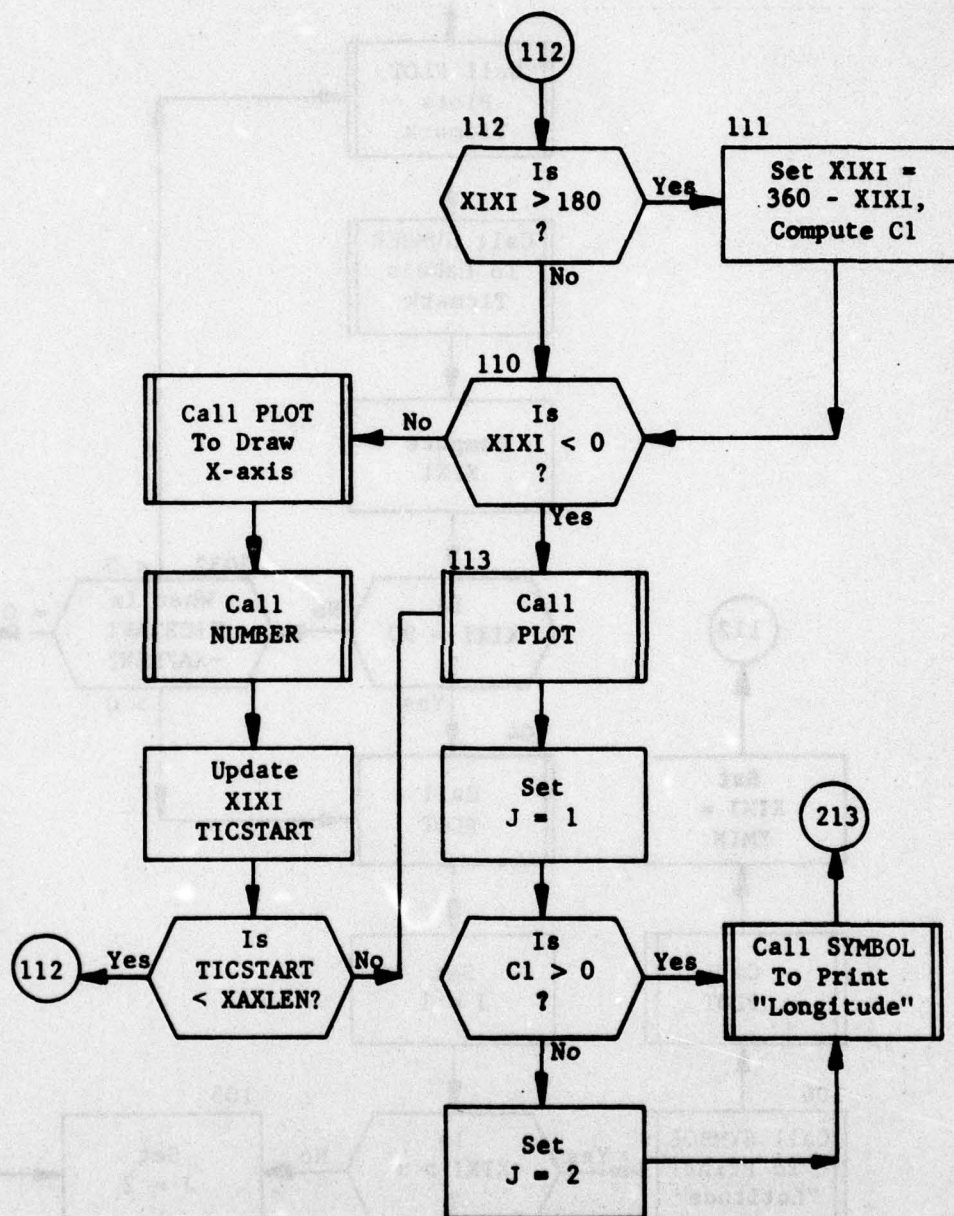


Figure 130.6. (Part 4 of 13)

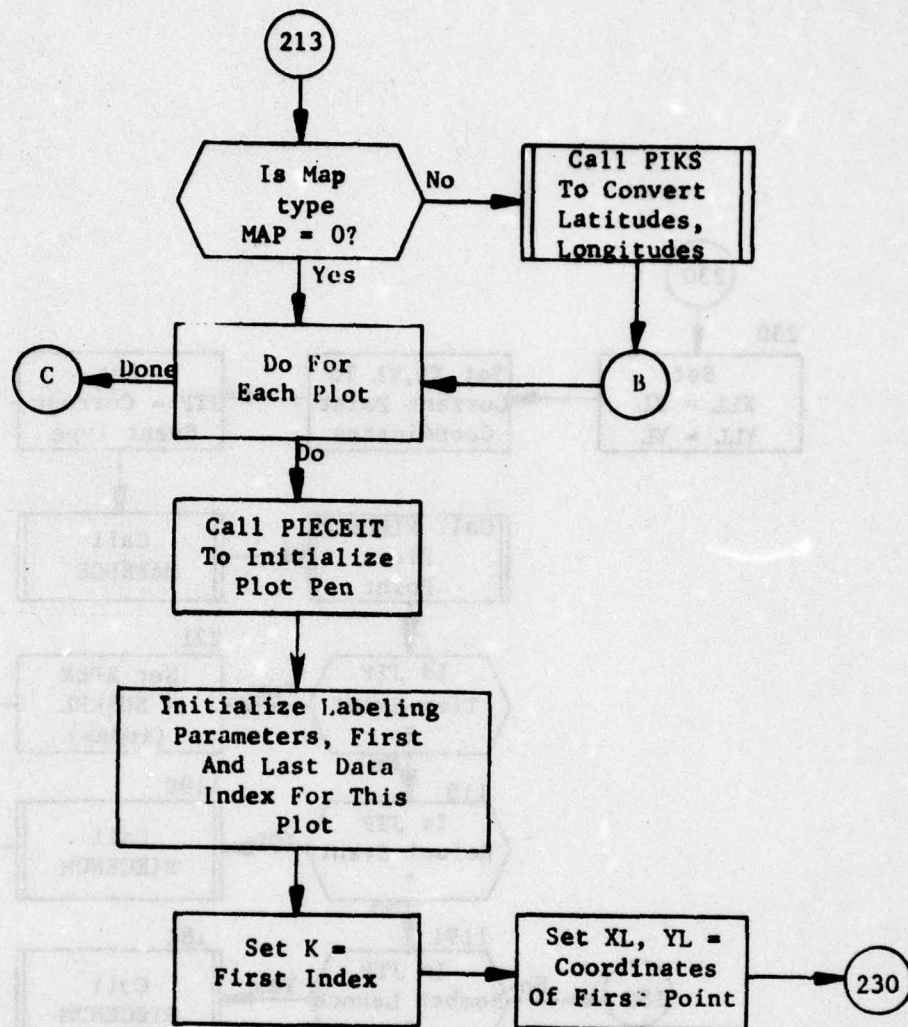


Figure 130.6. (Part 5 of 13)

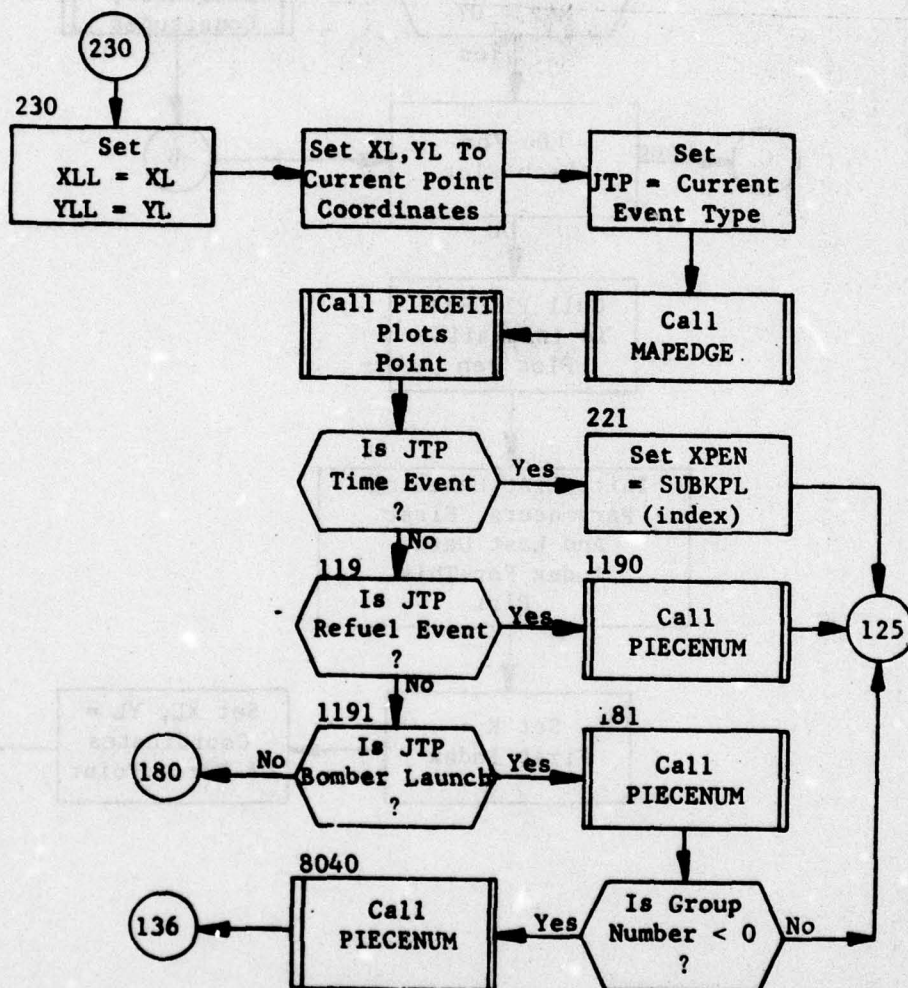


Figure 130.6. (Part 6 of 13)

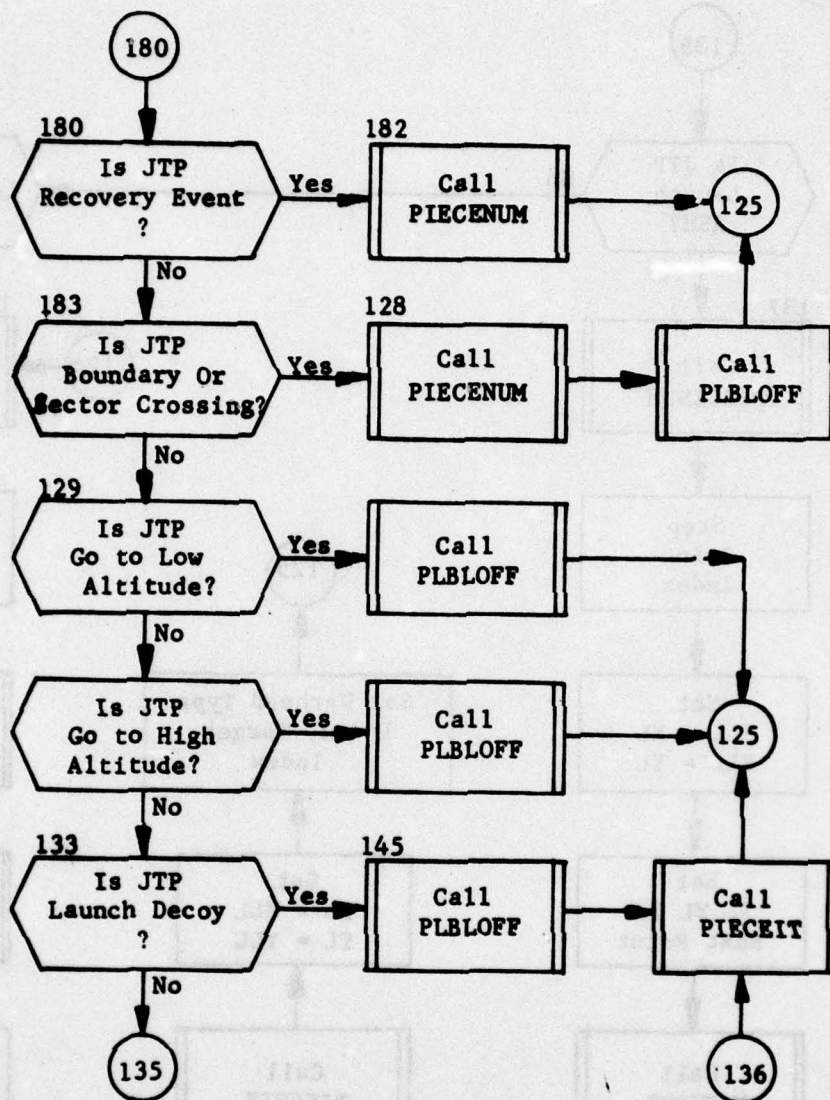


Figure 130.6. (Part 7 of 13)

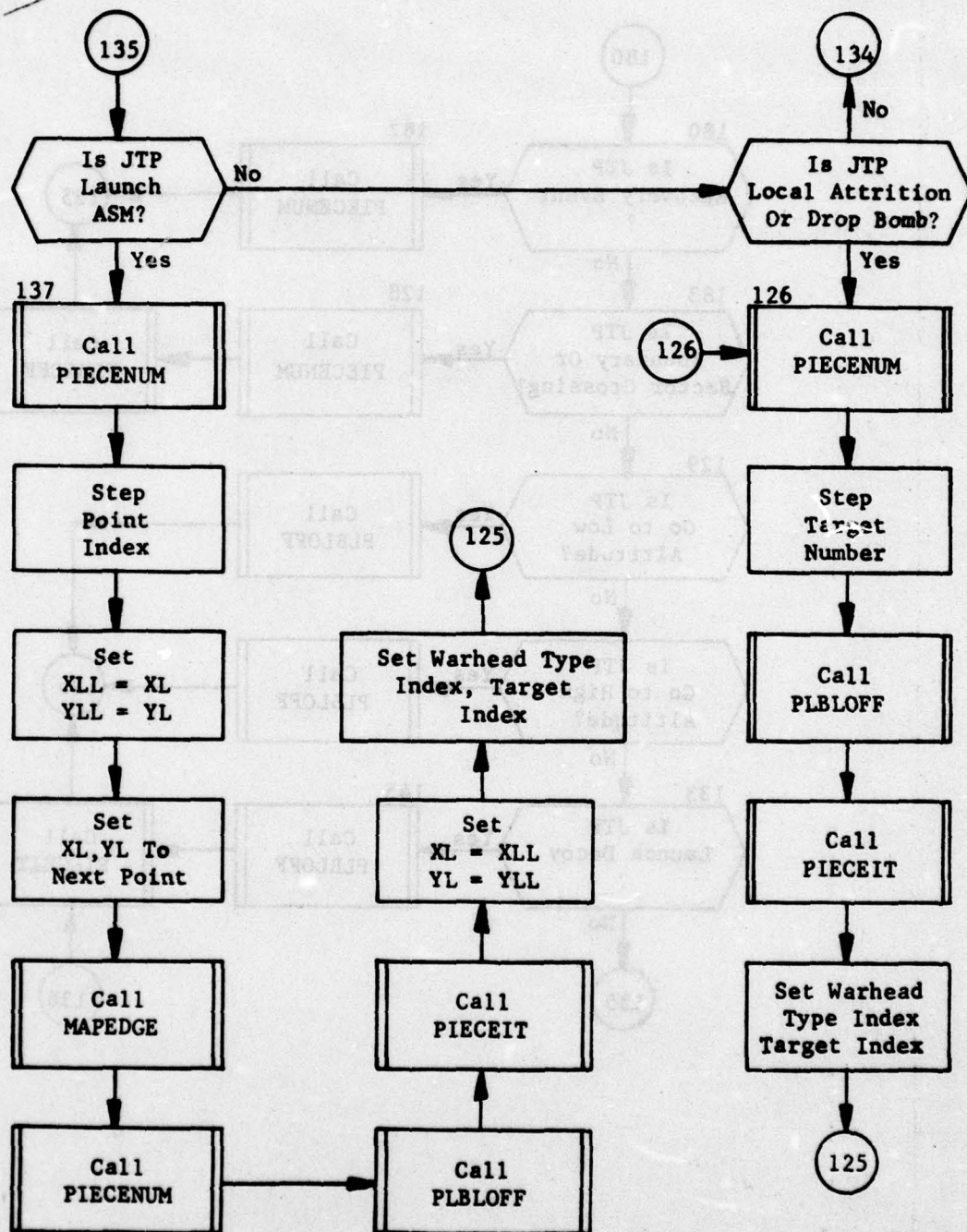


Figure 130.6. (Part 8 of 13)

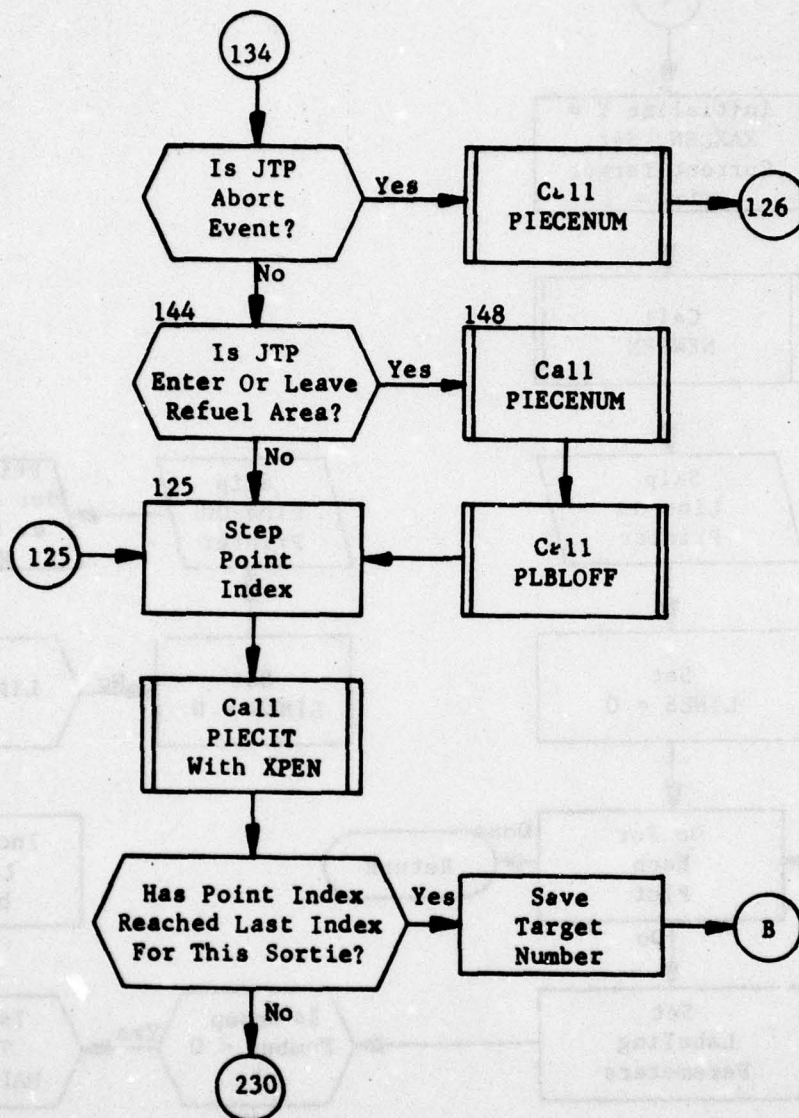


Figure 130.6. (Part 9 of 13)

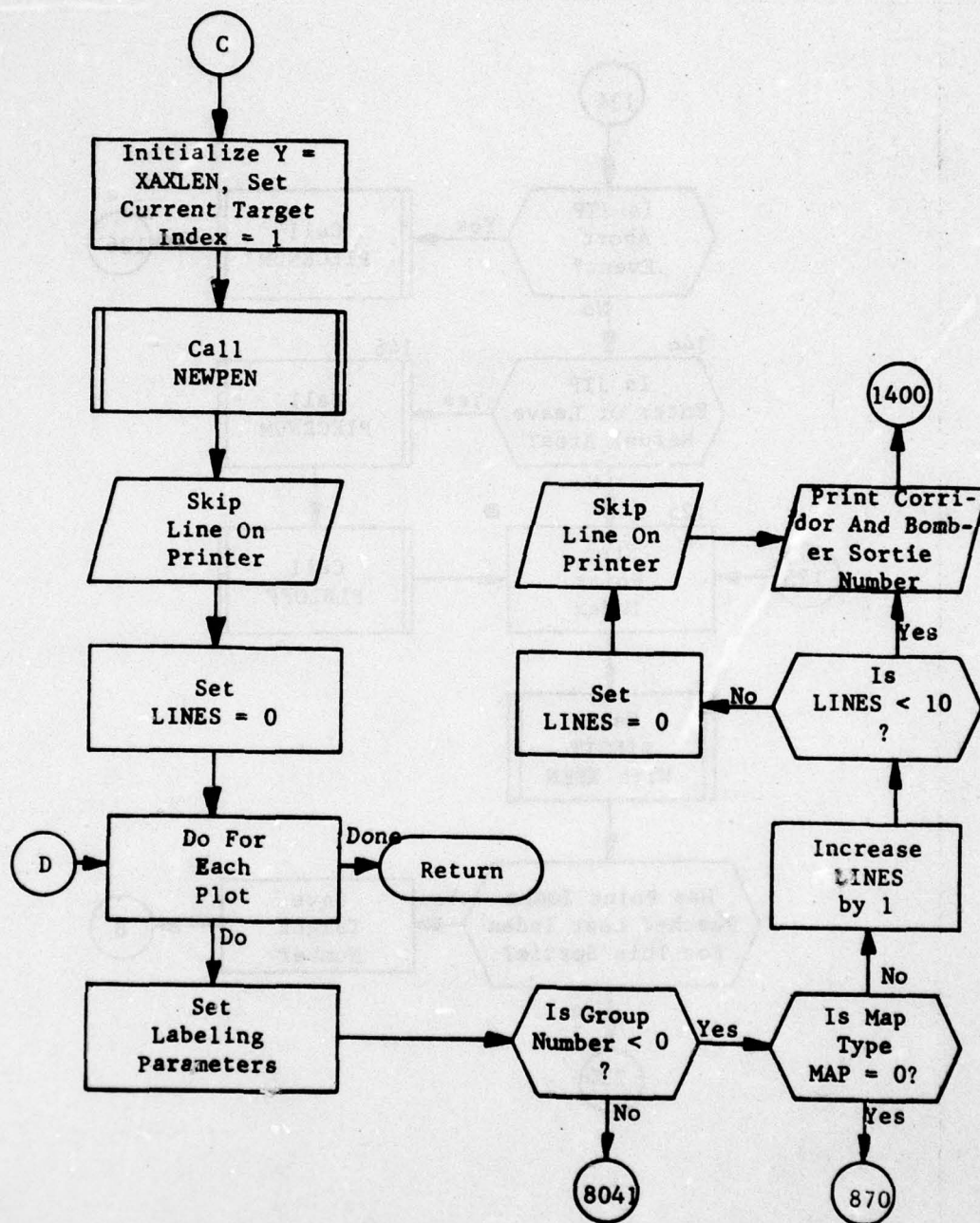


Figure 130.6. (Part 10 of 13)

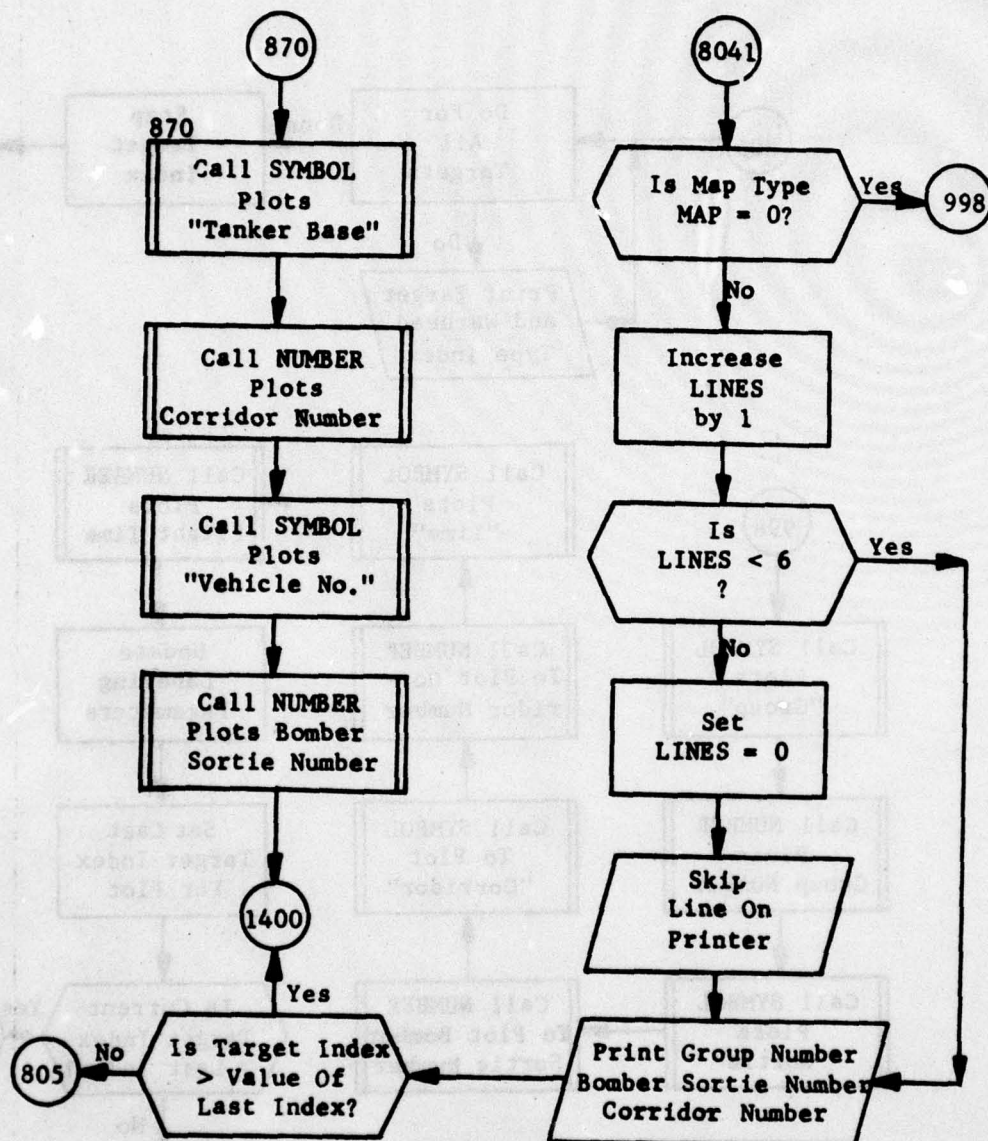


Figure 130.6. (Part 11 of 13)

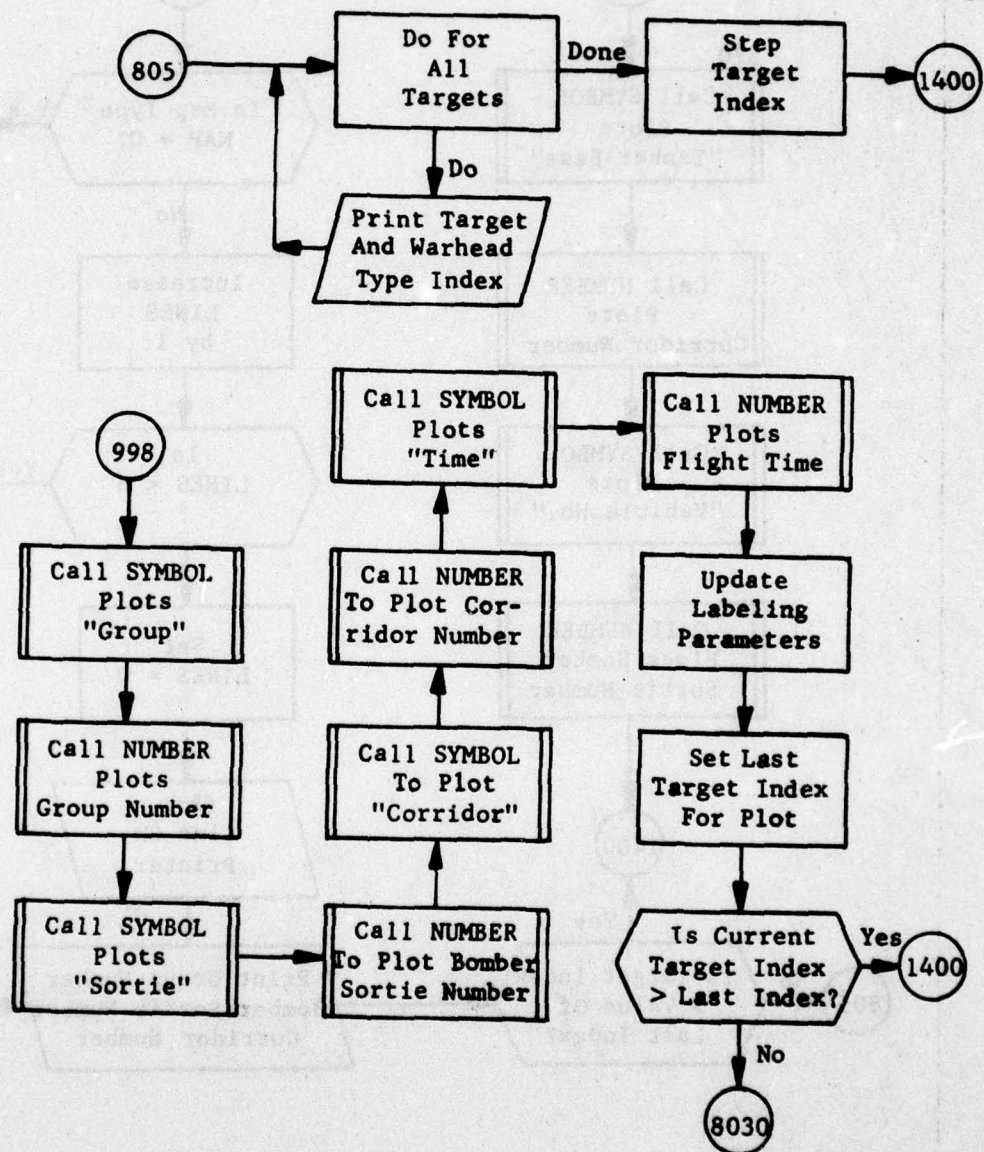


Figure 130.6. (Part 12 of 13)

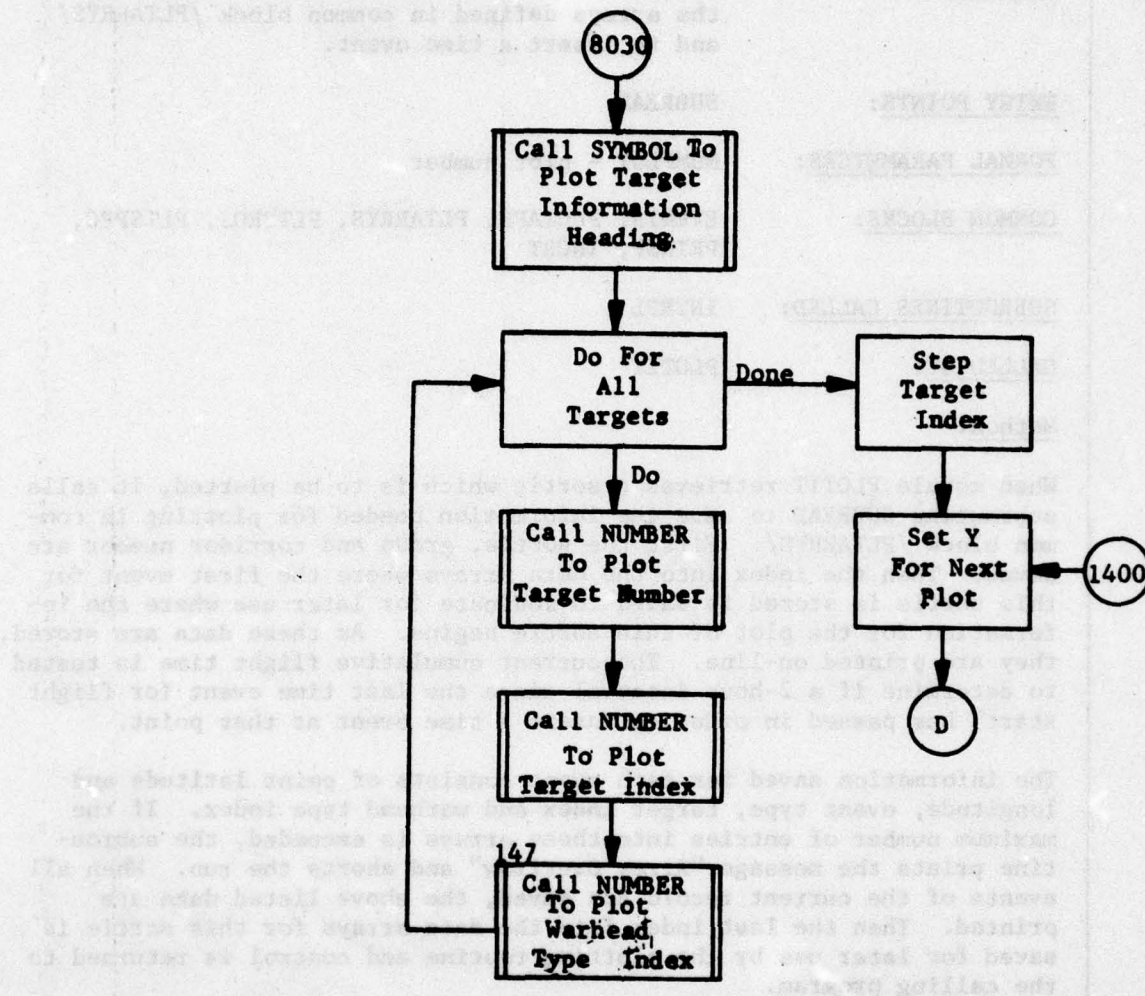


Figure 130.6. (Part 13 of 13)

8.12.6 Subroutine SUBREAD

PURPOSE: To save the information needed for plotting in the arrays defined in common block /PLTARRYS/ and to insert a time event.

ENTRY POINTS: SUBREAD

FORMAL PARAMETERS: NUMPLOT - plot number

COMMON BLOCKS: EVENTS, PLNTAPE, PLTARRYS, PLTPROJ, PLTSPEC, PRINSP, INCRT

SUBROUTINES CALLED: INTRPL

CALLED BY: PLOTIT

Method:

When module PLOTIT retrieves a sortie which is to be plotted, it calls subroutine SUBREAD to save the information needed for plotting in common block /PLTARRYS/. First the sortie, group and corridor number are saved. Then the index into the data arrays where the first event for this sortie is stored is saved to indicate for later use where the information for the plot of this sortie begins. As these data are stored, they are printed on-line. The current cumulative flight time is tested to determine if a 2-hour interval since the last time event (or flight start) has passed in order to insert a time event at that point.

The information saved for each event consists of point latitude and longitude, event type, target index and warhead type index. If the maximum number of entries into these arrays is exceeded, the subroutine prints the message "Array Overflow" and aborts the run. When all events of the current record are saved, the above listed data are printed. Then the last index into the data arrays for this sortie is saved for later use by the plotting routine and control is returned to the calling program.

Subroutine SUBREAD is illustrated in figure 130.7.

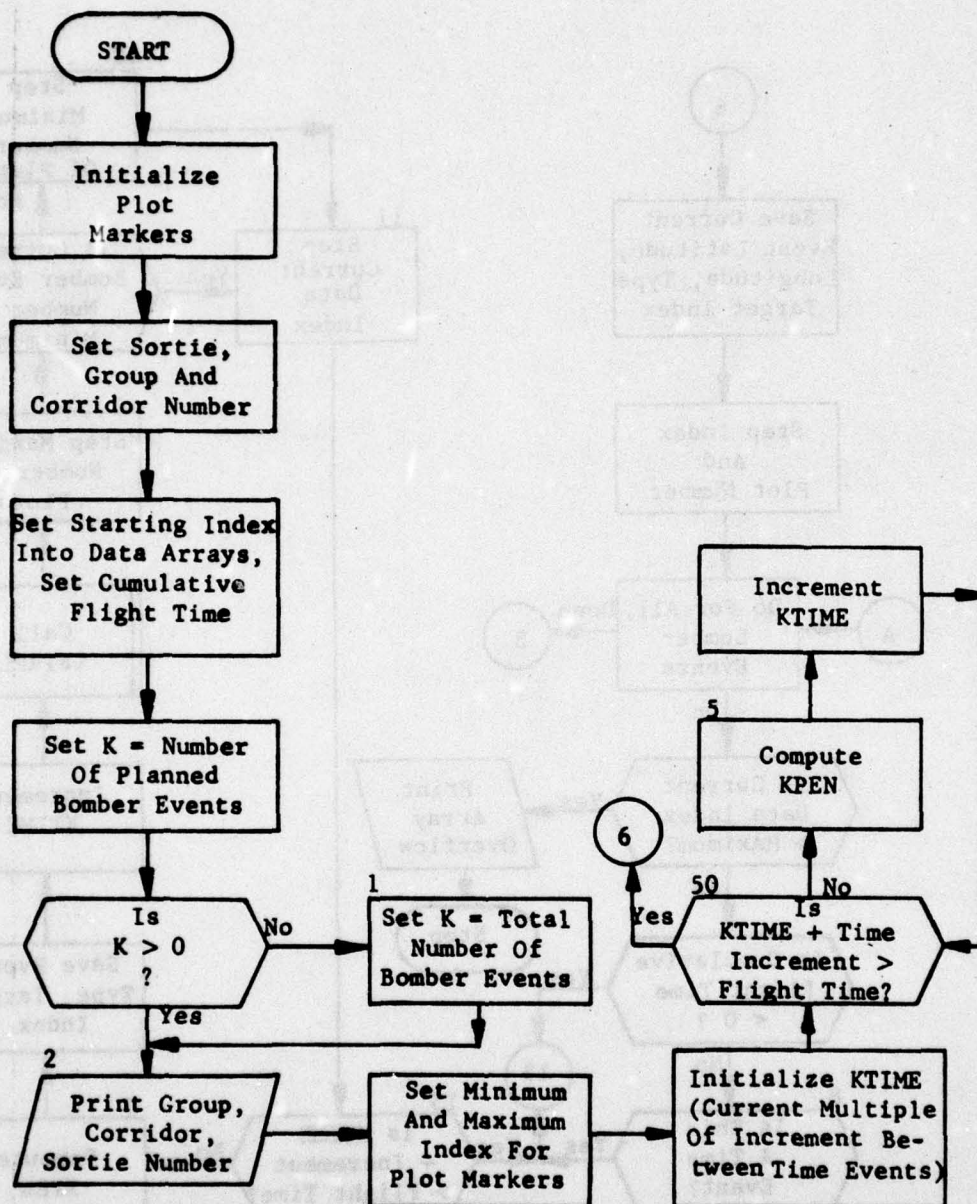


Figure 130.7. Subroutine SUBREAD (Part 1 of 3)

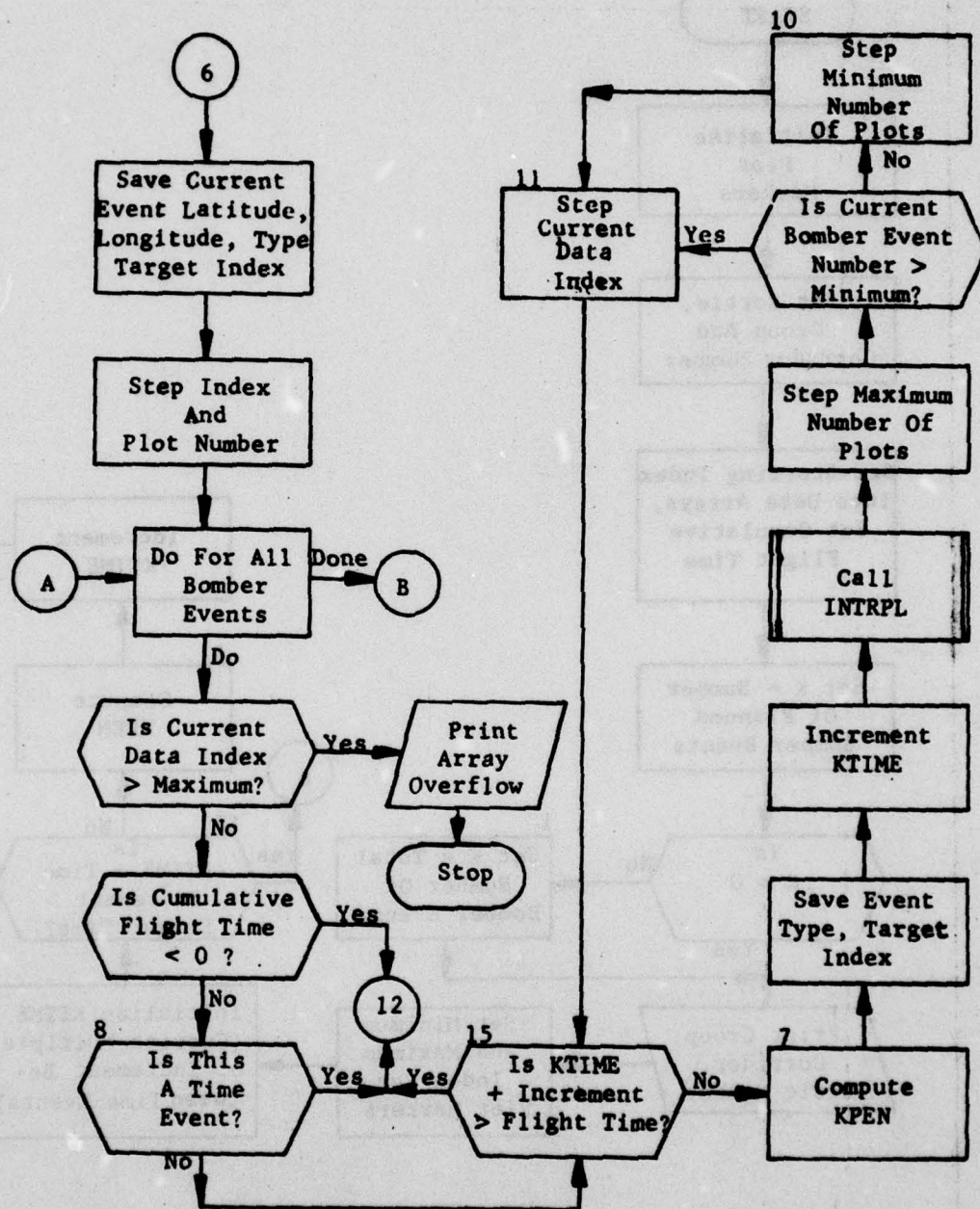


Figure 130.7. (Part 2 of 3)

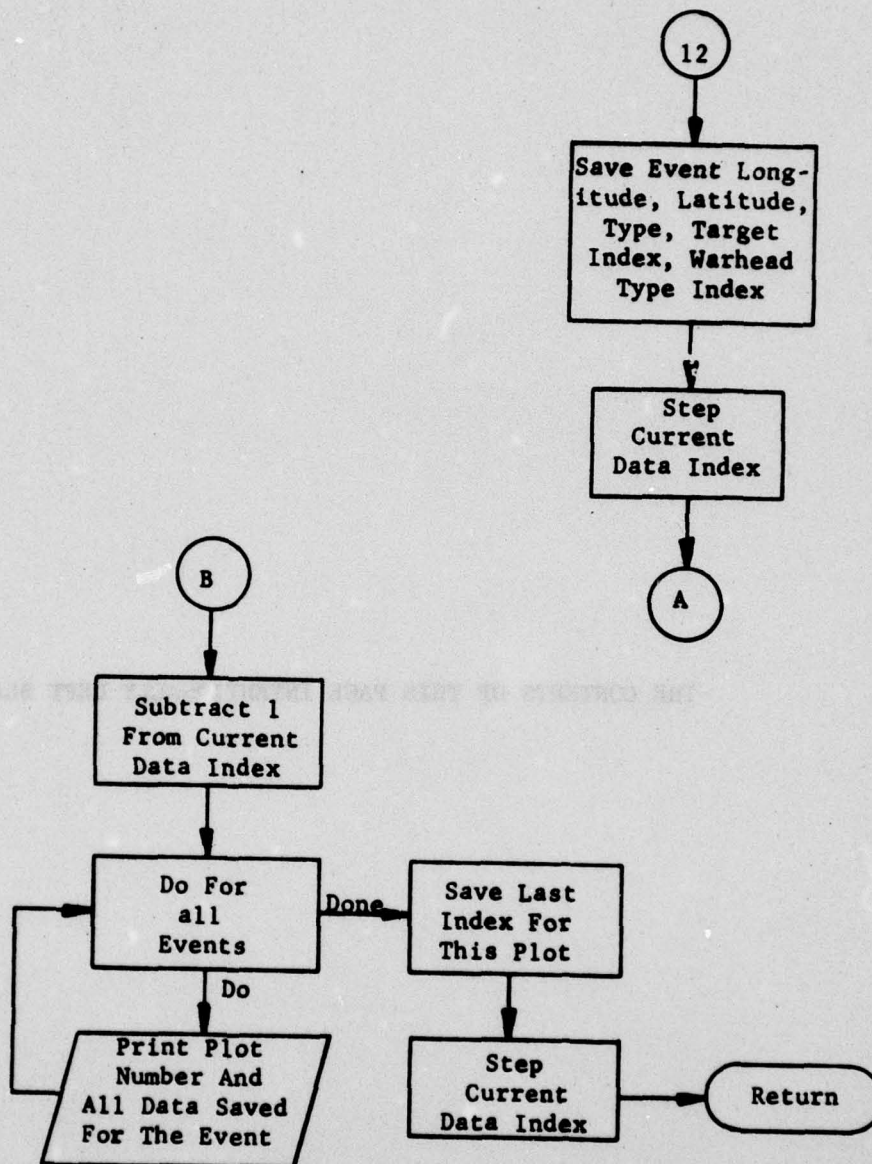


Figure 130.7. (Part 3 of 3)

9.16 Subroutine GETSTR*

PURPOSE: To obtain the next input string and make first interpretation

ENTRY POINTS: GETSTR

FORMAL PARAMETERS: None

COMMON BLOCKS: IPQT, STRING

SUBROUTINES CALLED: None

CALLED BY: ERRFND, INICOP, INPRIN, SRM

Method:

This subroutine which is a character-by-character analysis of an input string from a card image is best understood by reference to figure 145. Basically, the process attempts to discriminate between alphabetic and numeric data. Special characters which appear in array OPRATR also are discriminated. A string is terminated by a blank or a member of OPRATR.

*Note that this subroutine resides in the Main overlay of COP.

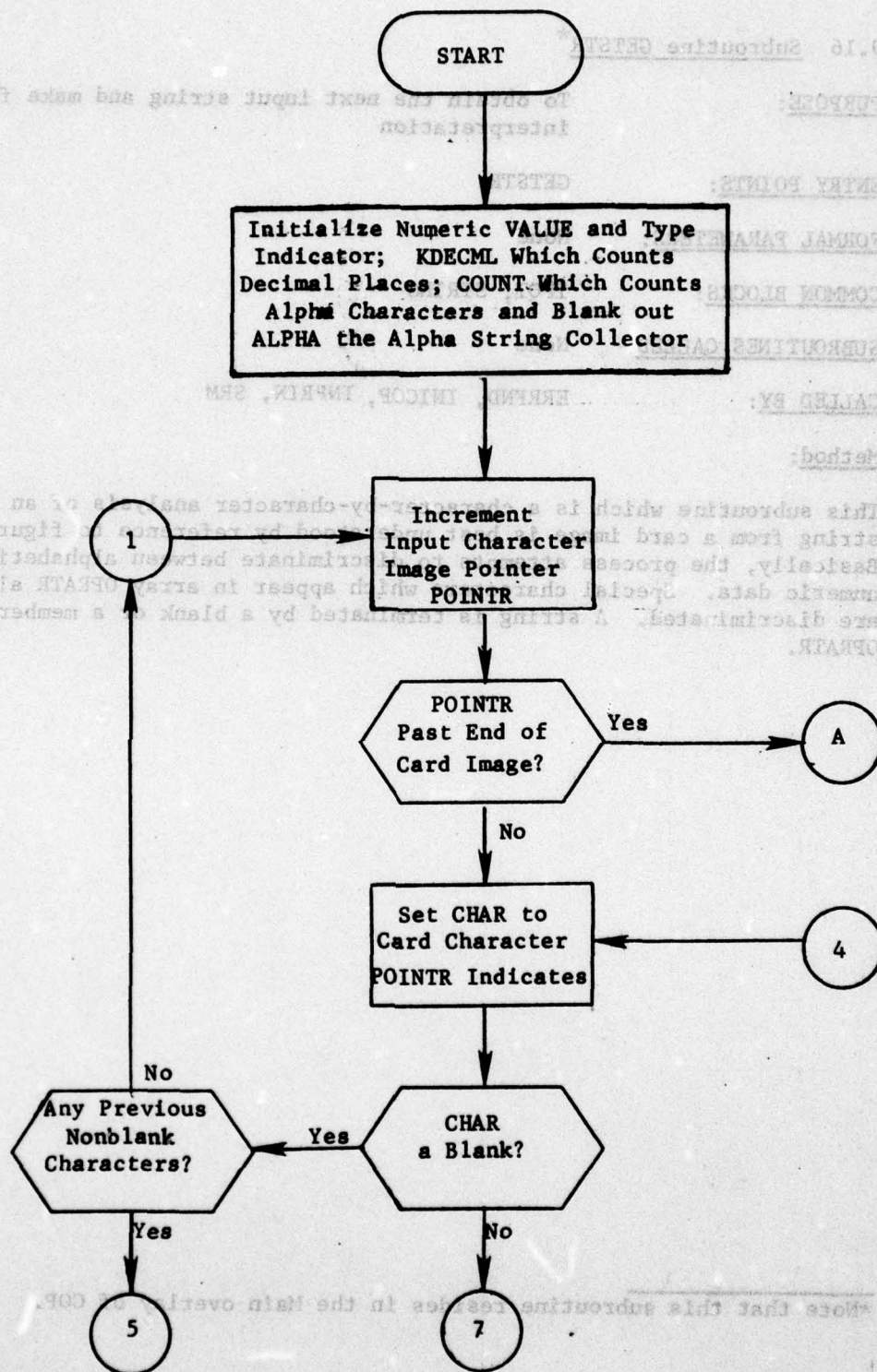


Figure 145. Subroutine GETSTR (Part 1 of 6)

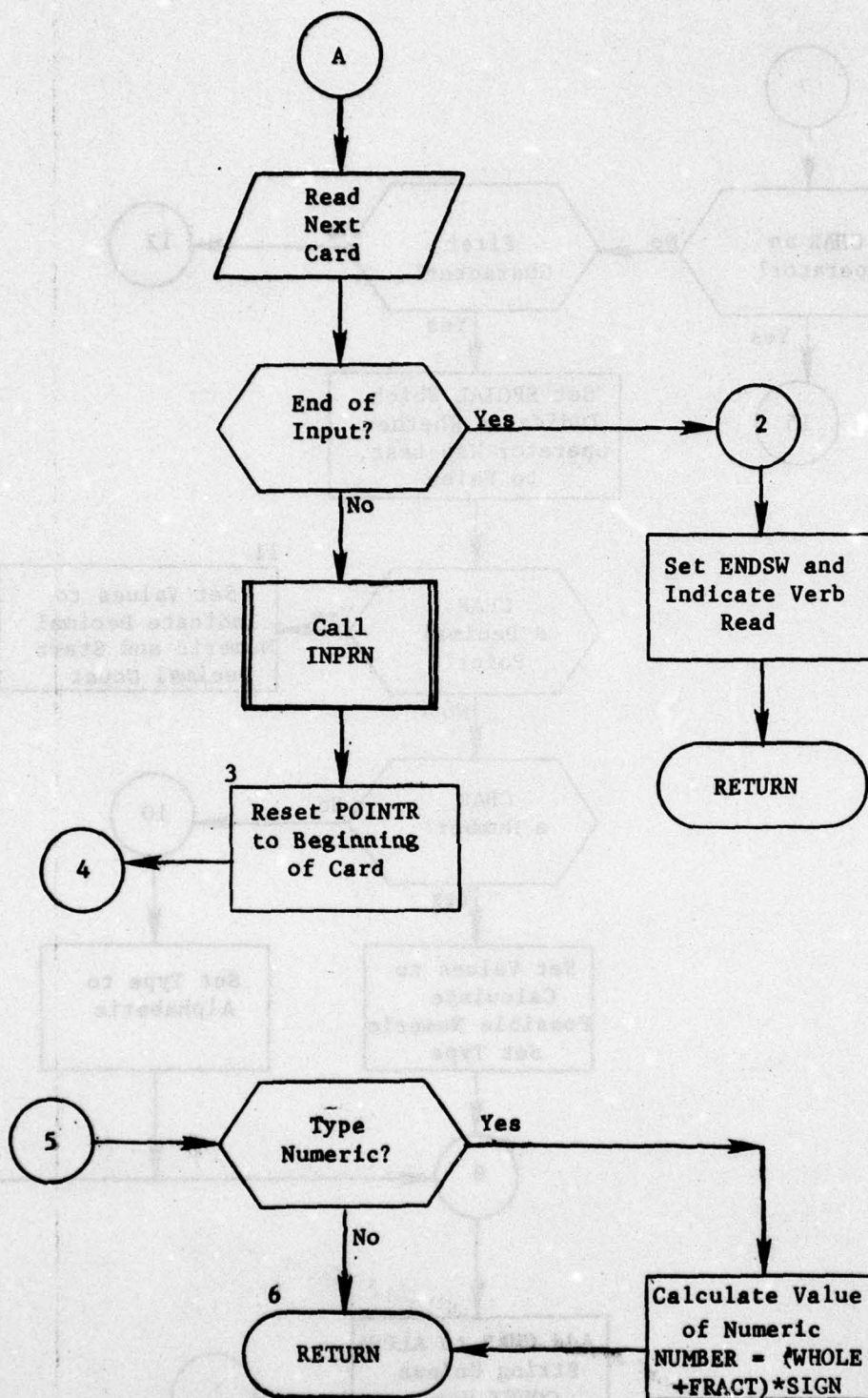


Figure 145. (Part 2 of 6)

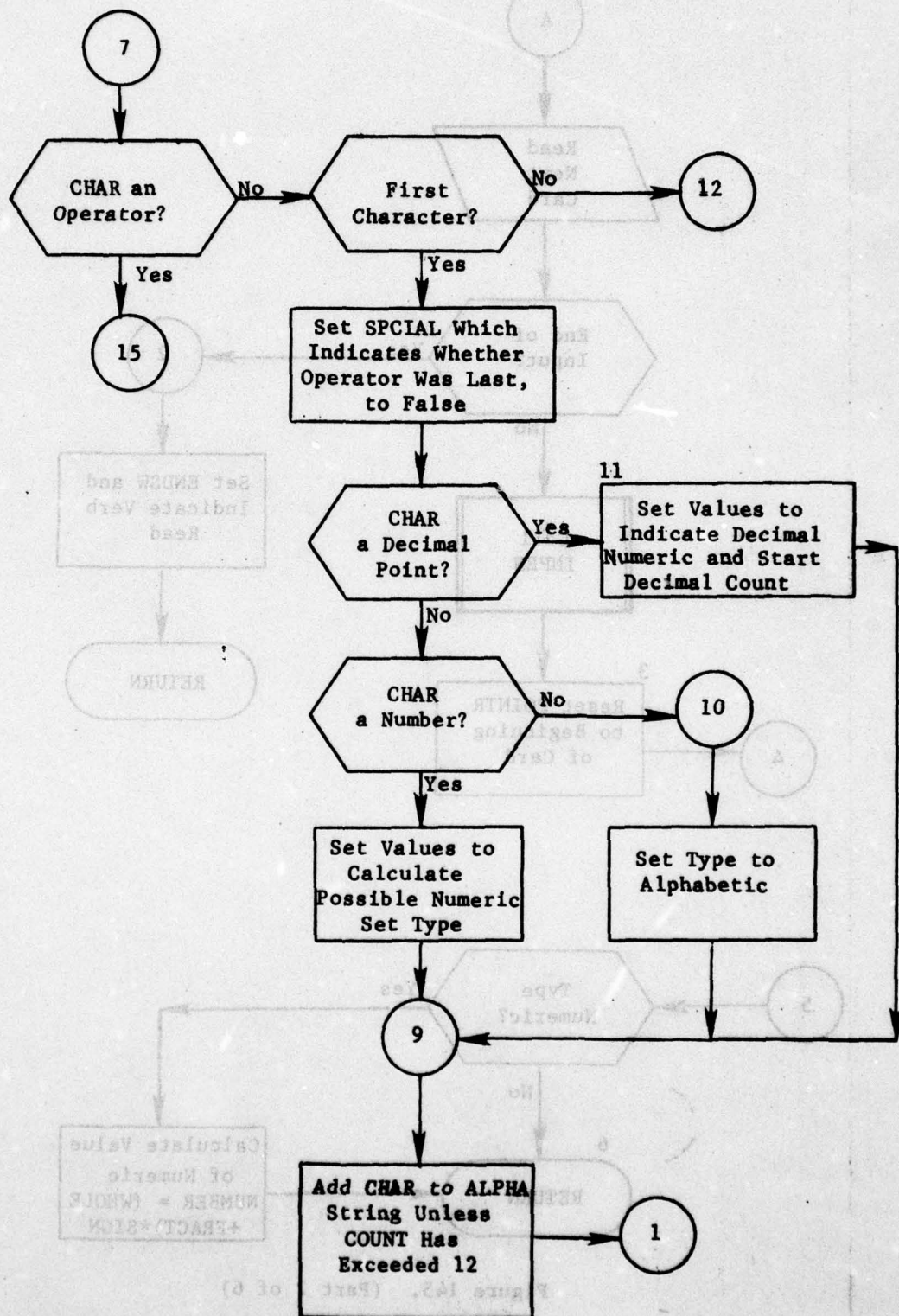


Figure 145. (Part 3 of 6)

9.21 Function IGETHOB

PURPOSE: To calculate actual weapon height of burst.

ENTRY POINT: IGETHOB

FORMAL PARAMETERS: IHOB - desired height of burst
ICNCD - vulnerability number
IYLD - weapon yield
IN - nonzero if called from PLANSET

COMMON BLOCKS: None

SUBROUTINE CALLED: None

CALLED BY: CALCOMP, INTRFACE

Method:

This function returns the actual weapon height of burst (in hundreds of feet) in R format if called from program PLANOUT. That is, the HOB is contained in the last three characters of the returned value in BCD code. The scaled height of burst is returned if called from program PLANSET. The formal parameter IHOB is an integer variable set to zero for ground burst and one for optimal air burst. The vulnerability number is contained as the last four BCD characters of formal parameter ICNCD if called from PLANOUT. (The country location code CNTRYLOC is the first two characters of this parameter.) The vulnerability is contained in the first four BCD characters of ICNCD if called from PLANSET. The weapon yield in kilotons is contained in formal parameter IYLD in R5 format (right-justified with zero fill to left).

The method of computing actual HOB is described in the Analytical Manual, Volume II, Chapter 2, Calculation of Actual Height of Burst. Function IGETHOB merely implements that calculation. The code for calculating the adjusted vulnerability number is very similar to that used in subroutine VLRADP of program PLANSET. Function KNOBLANK is used to convert the actual height of burst from integer hundreds of feet to BCD code in R format.

The input vulnerability code is decoded into the appropriate vulnerability number VN, the letter ("P" or "Q"), and the K-factor XK. The cube root of the yield in megatons is used to calculate the adjusted vulnerability number AVN. The scaled height burst SHOB is determined by a series of IF statements which use the table specified in the Analytical Manual as referenced in the preceding paragraph. Finally, the actual height of burst (AHOB) is calculated by multiplying by the scaled yield.

Figure 150 illustrates function IGETHOB.

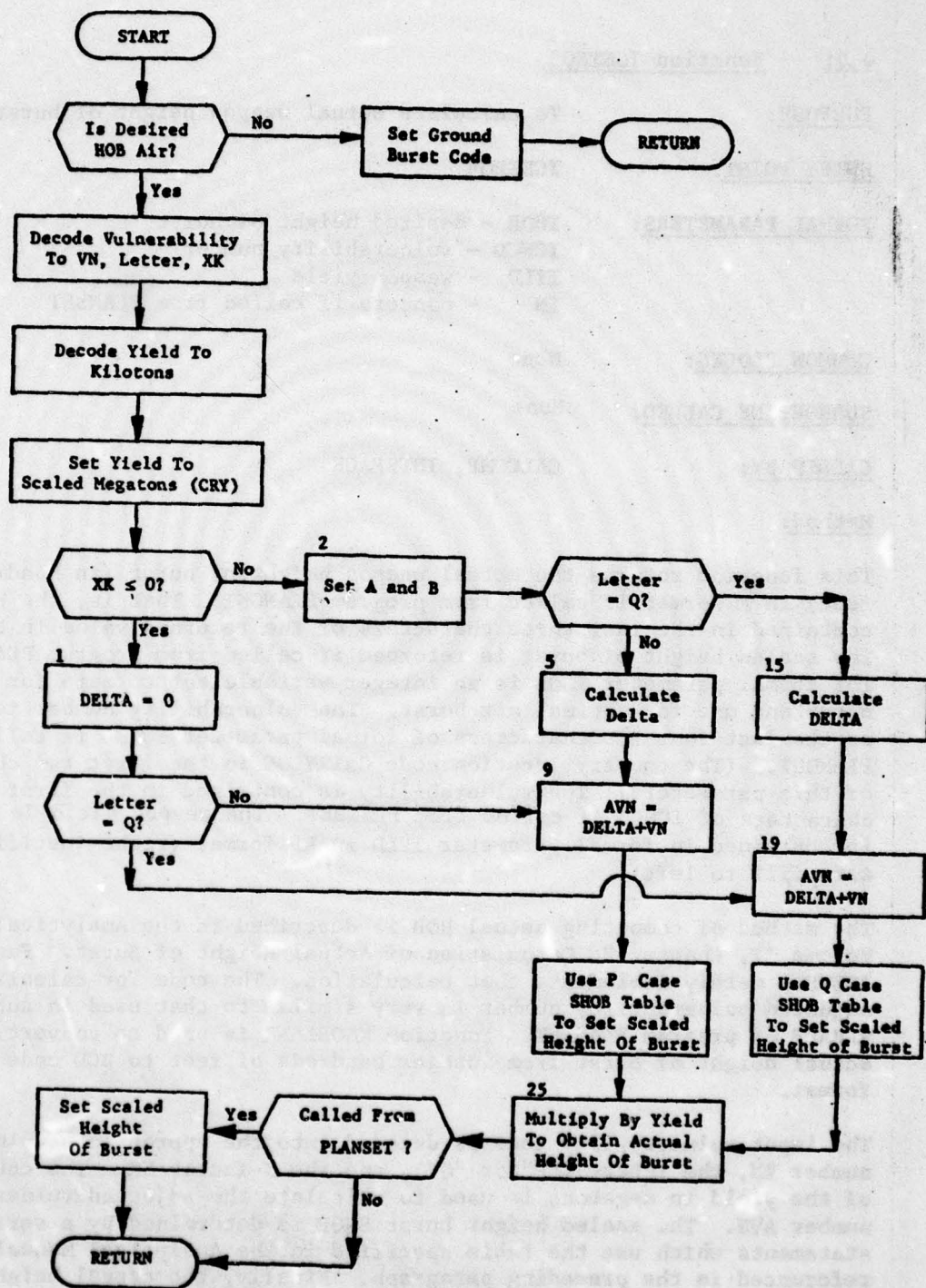


Figure 150. Function IGETHOB

AD-A061 096

COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM, (QUICK). PRO--ETC(U)
1978

F/G 5/2

UNCLASSIFIED

CCTC-CSM-MM-9-77-V1-CHG-1

NL

3 OF 3
ADA
061096

END

END
DATE
FILMED

2 -79
DDC

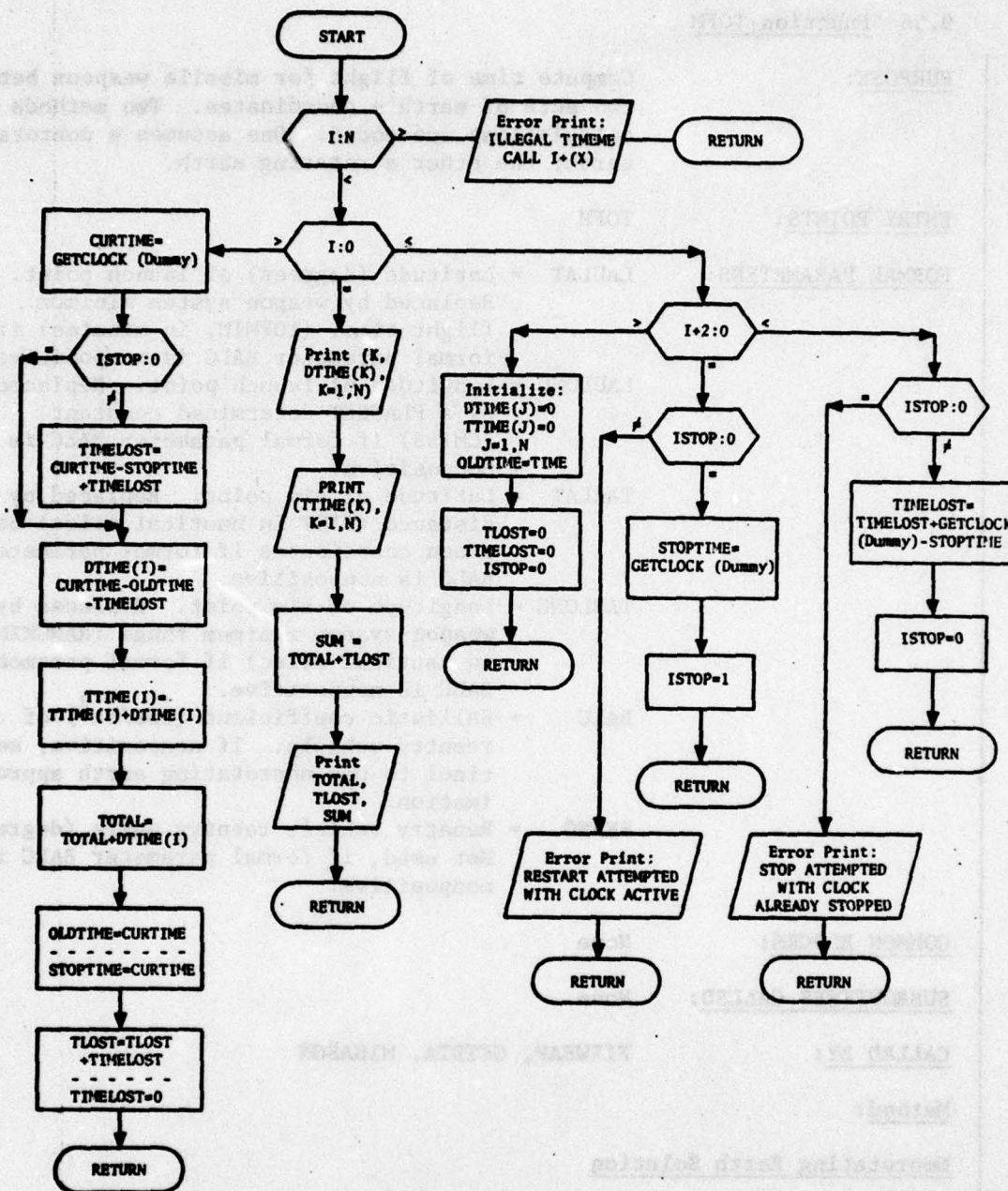


Figure 185. Subroutine TIMEME

9.56 Function TOFM

PURPOSE:

Compute time of flight for missile weapons between two sets of earth's coordinates. Two methods of calculations are coded. One assumes a nonrotating earth; the other a rotating earth.

ENTRY POINTS:

TOFM

FORMAL PARAMETERS:

LAULAT = Latitude (degrees) of launch point. Replaced by weapon system minimum flight time, (TOFMIN, in minutes) if formal parameter BALC is nonpositive

LAULONG = Longitude of launch point. Replaced by a PLANSET determined constant (CMISS) if formal parameter BALC is nonpositive.

TARLAT = Latitude of aim point. Replaced by distance (DIST in nautical miles) between coordinates if formal parameter BALC is nonpositive.

TARLONG = Longitude of aim point. Replaced by weapon system minimum range (RANGMIN in nautical miles) if formal parameter BALC is nonpositive.

BALC = Ballistic coefficient (lbs/ft^2) of reentry vehicle. If nonpositive, sentinel to use nonrotating earth approximation.

REANG = Reentry vehicle reentry angle (degrees). Not used, if formal parameter BALC is nonpositive.

COMMON BLOCKS:

None

SUBROUTINES CALLED:

None

CALLED BY:

FIXWEAP, GETDTA, MISASGN

Method:

Nonrotating Earth Solution

If the fifth formal parameter is nonpositive, time of flight is calculated assuming a nonrotating earth. Time of flight for a missile weapon is calculated based on distance between launch and target (DIST), constant (CMISS), and attributes TOFM and RANGMIN. Constant CMISS is computed for each missile type using attributes RANGE, TOFMIN, RANGMIN, and SPEED.

$$CMISS = \frac{(RANGE/SPEED) - TOFMIN}{(RANGE - RNGMIN)} \cdot 9$$

Time of flight, then equals

$$TOFMIN + CMISS (DIST - RNGMIN) \cdot 9$$

Rotating Earth Solution

If a ballistic coefficient is supplied, the fifth formal parameter, time of flight is determined under the assumption of a rotating earth.

The data needed for operation is as follows:

- o Launch location for booster (PX: Longitude; PY: Latitude)
- o Warhead data for each reentry object as follows:
 - Reentry angle (REANG); if no reentry angle is given, a minimum energy path is assumed
 - Ballistic coefficient (BALC) of reentry vehicle
 - Location of the target point (QX: Longitude; QY: Latitude)

The process results are a set of path segments that describe the movement of a launch vehicle and its subsidiaries from launch to target hit as follows:

- o Launch: Launch to orbit insertion as a linear path with constant acceleration. Orbit insertion is assumed at the atmosphere limit (ATML). The time to complete this path is called TB.
- o Elliptical Orbit: Elliptical orbit commensurate with a minimum energy trajectory or a specified angle to reentry into sensible atmosphere. The time to complete this path is called TF.
- o Reentry to deceleration band: The first reentry segment is a linear path with constant velocity to the point where deceleration becomes significant. The time to complete this path is called TR1.
- o Deceleration: The second reentry segment is a linear path with deceleration to the point where terminal velocity approached. The time to complete this path is called TR2.
- o Terminal phase: The third reentry segment is a linear path with constant velocity to the point of detonation. The time to complete this path is called TR3.

The elliptical orbit path generation consumes the vast amount of the calculation. This path is calculated initially followed by the various paths defined within the earth's atmosphere. Kepler's equations are used for the elliptical orbit determination along with a heuristic cycling approach. The main problem is the earth's rotation since as the reentry vehicle spends time (the unknown parameter to be determined) above the atmosphere, the longitude of the aim point is moving and the amount of movement is dependent on the TOF. Iterative approaches resolve the dilemma. The aim point longitude is calculated using a TOF from a previous set of calculations (initially set to zero) and a new TOF is determined based on the given longitude. If the new TOF compares within tolerances (say 5 seconds) of the previously determined TOF solution is final. If the two TOF are outside tolerances, the new TOF is used to calculate the aim point longitude and equations are solved anew. This iterative approach converges for all possible trajectories.

In addition to defined parameters the following parameters are necessary for calculations:

TTO: Total TOF on a previous interaction (initiated to zero)
 ERAT: Earth's rotational rate
 ACON: Earth's radius
 G: One over the square root of the gravitational constant

Elliptical Orbit Determination: Begin calculations of elliptical parameters by computing great circle range angle 'ALPHO2' between launch and target points as:

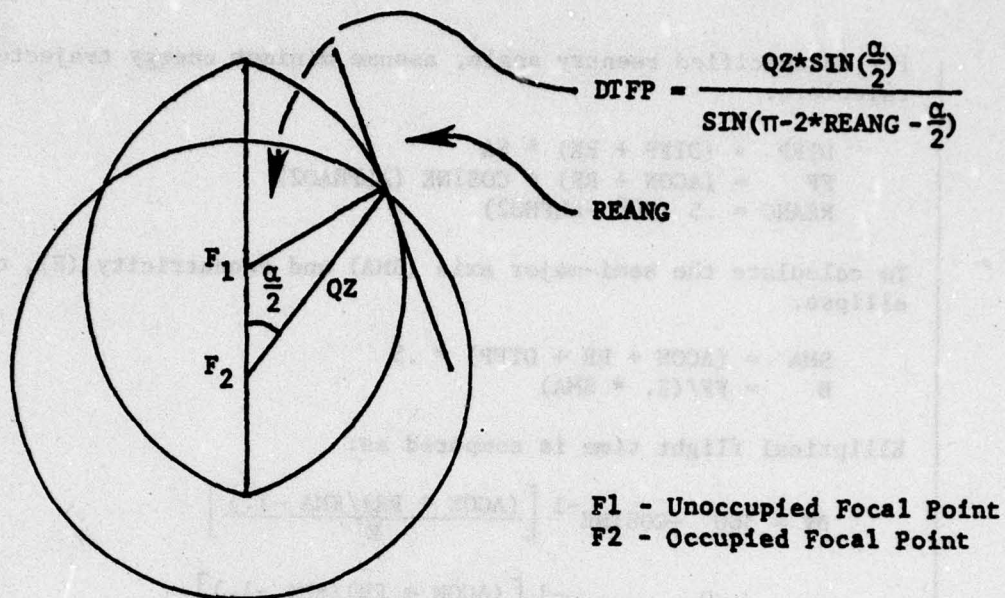
SY = SINE (PY)
 QX = QX + ERAT (TF -TTO)
 TTO = TF
 SW = SY * SINE (QY)
 COSA = WQ * COSINE (PX -QX) + COSINE (PY) * COSINE (QY)
 ALPHAX = COS⁻¹ (COSA)
 ALPHO2 = ALPHAX * .5
 SA = SIN (ALPHO2)

Compute distance to unoccupied focal point (DTFP) from target point in plane of ellipse, and distance between focal points (FF). Calculations assume a synetrical ellipse about apogee of target and launch, see figure 186.

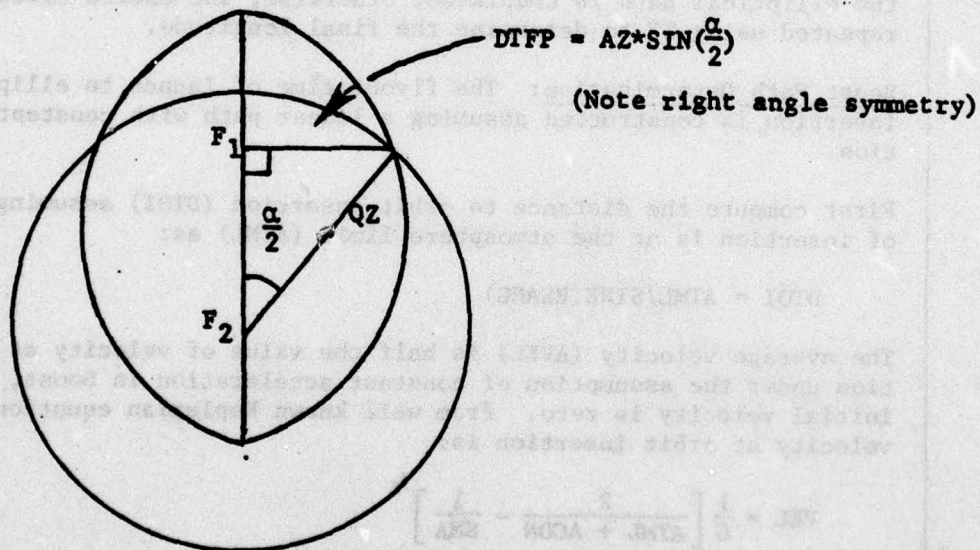
For a specified reentry angle calculate:

DTFP = (ACON + RR) * SA/SINE (2.*REANG + ALPHO2)
 FF = (DTFP * SINE (2. * REANG)/SA

where RR is the altitude of the aim point.



Case 1 -- Reentry Angle Specified



Case 2 -- Minimum Energy Specified

Figure 186. Focal Point Calculations

For no specified reentry angle, assume minimum energy trajectory and calculate:

$$\begin{aligned} \text{DTFP} &= (\text{DTFP} + \text{RR}) * \text{SA} \\ \text{FF} &= (\text{ACON} + \text{RR}) * \text{COSINE} (\text{ALPHA02}) \\ \text{REANG} &= .5 (90^\circ - \text{ALPHA02}) \end{aligned}$$

To calculate the semi-major axis (SMA) and eccentricity (E), of the ellipse.

$$\begin{aligned} \text{SMA} &= (\text{ACON} + \text{RR} + \text{DTFP}) * .5 \\ \text{E} &= \text{FF} / (2. * \text{SMA}) \end{aligned}$$

Elliptical flight time is computed as:

$$\begin{aligned} \text{BV} &= 360^\circ - \text{COSINE}^{-1} \left[\frac{(\text{ACON} + \text{RR}) / \text{SMA} - 1.}{\text{E}} \right] \\ \text{FV} &= 360^\circ + \text{COSINE}^{-1} \left[\frac{(\text{ACON} + \text{RR}) / \text{SMA} - 1.}{\text{E}} \right] \\ \text{TF} &= \text{G} * \text{SMA}^{3/2} \left[(\text{FV} - \text{BV} + \text{E} * (\text{SINE} (\text{FV}) - \text{SINE} (\text{BV}))) \right] \end{aligned}$$

This time of flight (TF) is compared with the last calculated time of flight (TTO). If the difference is within tolerances, say 5 seconds, the elliptical path is completed; otherwise, the entire calculations are repeated using TF to determine the final longitude.

Boost Path Determination: The flyout time of launch to elliptical orbit insertion is constructed assuming a linear path with constant acceleration.

First compute the distance to orbit insertion (DTOI) assuming altitude of insertion is at the atmosphere limit (ATML) as:

$$\text{DTOI} = \text{ATML} / \text{SINE}(\text{REANG})$$

The average velocity (AVEL) is half the value of velocity at orbit insertion under the assumption of constant acceleration in boost, and that initial velocity is zero. From well known Keplerian equations, the velocity at orbit insertion is:

$$\text{VEL} = \frac{1}{\text{G}} \left[\frac{2}{\text{ATML} + \text{ACON}} - \frac{1}{\text{SMA}} \right]^{1/2}$$

Therefore the time to orbit insertion is

$$\begin{aligned} \text{AVEL} &= .5 * \text{VEL} \\ \text{TB} &= \text{DTOI} / \text{AVEL} \end{aligned}$$

Reentry Path Determination: Reentry time is now calculated by constructing three segments for each reentry object. The three segments are: constant velocity to the point where deceleration becomes significant; deceleration path to the point where terminal velocity is approach; segment of constant velocity to the aim point.

Start construction of the reentry segments from the altitude of maximum deceleration (AMD)

$$SA = \text{SINE}(\text{REANG})$$

$$AMD = \frac{22000.}{6080.} * \text{LOG} \left[.0034 * 32.2 * 22000 / (\text{BALC} * SA) \right]$$

Where BALC is the ballistic coefficient.

22000. scales the atmosphere density.

.0034 is the sea level density in slugs/FT³.

Define half the altitude width of the deceleration segment as:

$$\text{HWAB} = 50,000/6080.$$

A constant is used since this segment has little variance for different weapons systems. Time for first segment is now

$$\text{TR1} = (\text{ATML} - \text{AMD} - \text{HWAB}) / (\text{SA} * \text{VEL})$$

The velocity at the end of the second segment is computed as:

$$\text{CON} = \frac{-22000}{2} * 32.2 * .0034 / (\text{BALC} * \text{SA})$$

$$\text{VELT} = \text{VEL} * \text{EXP} \frac{\text{HWAB} - \text{AMD}}{22000.}$$

Second segment time interval then is:

$$\text{TR2} = \frac{4. * \text{HWAB}}{\text{SA}(\text{VELT} - \text{VEL})}$$

The last segment time interval is:

$$\text{TR3} = \frac{(\text{AMD} - \text{HWAB})}{\text{SA} * \text{VELT}}$$

Total trajectory time, then, is the summation of the defined time segments.

Function TOFM is illustrated in figure 186.1.

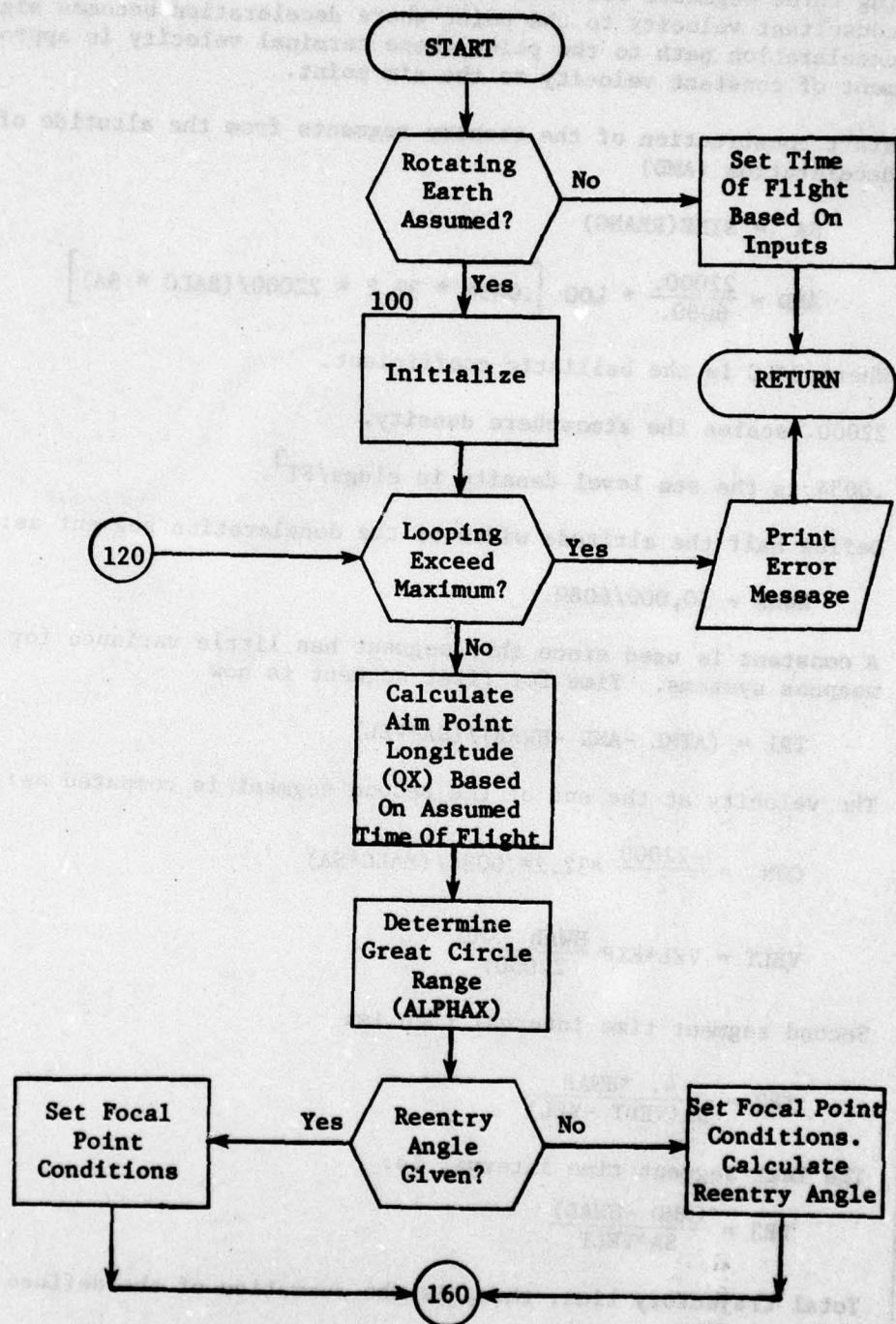
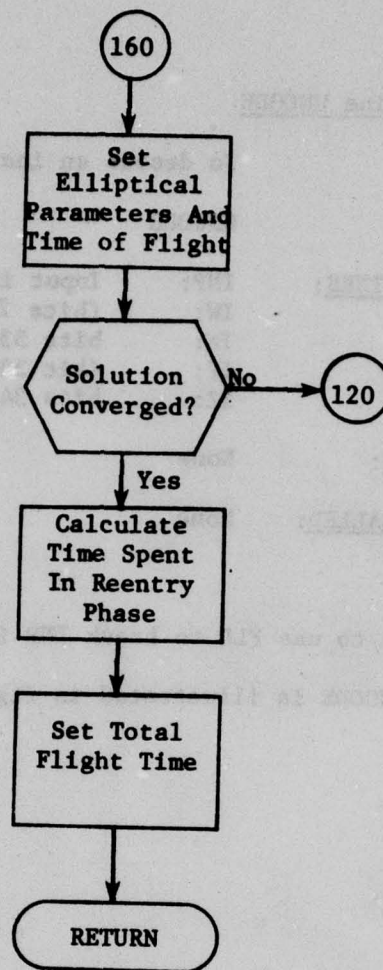


Figure 186.1. Function TOFM (Part 1 of 2)



(Part 2 of 2)

9.57 Subroutine UNCODE

PURPOSE:

To decode an instruction code

ENTRY POINTS:

UNCODE

FORMAL PARAMETERS:

INP: Input instruction code
IW: (bits 29-32) +1
IX: bits 33-35
IY: (bit 33) +1
IZ: bits 34-35

COMMON BLOCKS:

None

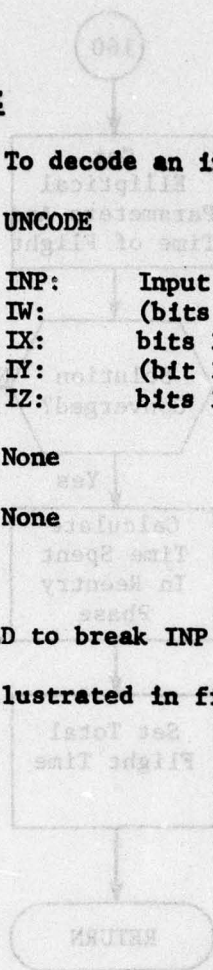
SUBROUTINES CALLED:

None

Method:

The method is to use FLD to break INP into IW, IX, IY, and IZ.

Subroutine UNCODE is illustrated in figure 187.



APPENDIX B

EXECUTABLE JOB CONTROL LANGUAGE (JCL) QUICK SYSTEM

The QUICK system executes from an H*(HIS6000 System Loadable file). Figure 194 contains the JCL necessary to build the H* directly from source files. Note that within the FORTY activity for each link an object file (catalog 632IDPOO/QUIK/OBJECT) is also created. Figure 195 contains the JCL to recreate the H* from these object files. In effect, the programmer needs only replace the SELECT of the object deck with the corresponding FORTY activity for each overlay link in which the source has changed to recreate the H*. Finally, figure 196 contains the JCL to create the QUICK utility library from source.

```

$      IDENT  5162,COMPL,314,I M A PROG,631,12
$      USERID 632IDP99$PASSWORD/UZZ
$      LIMITS 30,60K,,50K
$      OPTION FORTRAN,NOGO
$      LIBRARY UL,PL
$      LOWLOAD
$      ENTRY  .....
$      FORTY  MAP,XREF,DECK
$      PRMFL  C*,W,S,632IDP00/QUIK/OBJECT/COP
$      SELECT 632IDP00/QUIK/SOURCE/COP/COP
$      SELECT 632IDP00/QUIK/SOURCE/COP/BANNER
$      SELECT 632IDP00/QUIK/SOURCE/COP/ERPROC
$      SELECT 632IDP00/QUIK/SOURCE/COP/HDFND
$      SELECT 632IDP00/QUIK/SOURCE/COP/INICOP
$      SELECT 632IDP00/QUIK/SOURCE/COP/INPRIN/
$      SELECT 632IDP00/QUIK/SOURCE/COP/INPUT
$      SELECT 632IDP00/QUIK/SOURCE/COP/MODGET
$      IDS    DECK
$      LIMITS 30,60K,,50K
$      FILE   *3,X3R,100R
$      PRMFL  C*,W,S,632IDP00/QUIK/OBJECT/QDATA
$      SELECT 632IDP00/QUIK/SOURCE/COP/QDATA
$      SELECT 632IDP00/QUIK/SOURCE/COP/QDATB
$      USE     .QMAX/1/,.QAREA/3126/,.QMIN/1/,.FRRD.
$      LINK   BOOTT
$      FORTY  MAP,XREF,DECK
$      PRMFL  C*,W,S,632IDP00/QUIK/OBJECT/BOOT
$      SELECT 632IDP00/QUIK/SOURCE/BOOT
$      SELECT 632IDP00/QUIK/SOURCE/COP/DCTFND
$      SELECT 632IDP00/QUIK/SOURCE/COP/MNMFND
$      SELECT 632IDP00/QUIK/SOURCE/COP/NUMFND
$      SELECT 632IDP00/QUIK/SOURCE/COP/RNMFND
$      SELECT 632IDP00/QUIK/SOURCE/COP/SEEKER
$      SELECT 632IDP00/QUIK/SOURCE/COP/STRMAK
$      LINK   TABSTR,BOOT
$      USE     TABLZ/808/
$      LINK   ERRF
$      FORTY  MAP,XREF,DECK
$      PRMFL  C*,W,S,632IDP00/QUIK/OBJECT/ERRFND
$      SELECT 632IDP00/QUIK/SOURCE/COP/ERRFND
$      SELECT 632IDP00/QUIK/SOURCE/COP/LNGSTR
$      SELECT 632IDP00/QUIK/SOURCE/COP/SYNTAX
$      SELECT 632IDP00/QUIK/SOURCE/COP/TABINS
$      SELECT 632IDP00/QUIK/SOURCE/COP/WEBSTR
$      LINK   INPT,ERRF
$      FORTY  MAP,XREF,DECK

```

Figure 194. H* Creation From Source (Part 1 of 11)

```

$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/INPTRN
$ SELECT 632IDP00/QUIK/SOURCE/COP/INPTRN
$ SELECT 632IDP00/QUIK/SOURCE/COP/DELTAB
$ SELECT 632IDP00/QUIK/SOURCE/COP/INMATH
$ SELECT 632IDP00/QUIK/SOURCE/COP/LINEIO
$ SELECT 632IDP00/QUIK/SOURCE/COP/PARLEV
$ SELECT 632IDP00/QUIK/SOURCE/COP/TABGET
$ LINK JLM,TABSTR
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/JLM
$ SELECT 632IDP00/QUIK/SOURCE/JLM/JLM
$ LINK ASSI
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/ASSIGN
$ SELECT 632IDP00/QUIK/SOURCE/JLM/ASSIGN
$ SELECT 632IDP00/QUIK/SOURCE/JLM/ALPHAS
$ SELECT 632IDP00/QUIK/SOURCE/JLM/PLAYERS
$ SELECT 632IDP00/QUIK/SOURCE/JLM/TOPRINT
$ LINK SELE,ASSI
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/SELECT
$ SELECT 632IDP00/QUIK/SOURCE/JLM/SELECT
$ SELECT 632IDP00/QUIK/SOURCE/JLM/ADTOBASE
$ SELECT 632IDP00/QUIK/SOURCE/JLM/DEFAULT
$ SELECT 632IDP00/QUIK/SOURCE/JLM/KRUNCH
$ SELECT 632IDP00/QUIK/SOURCE/JLM/SAMSET
$ LINK ASTE,SELE
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/ASTERISK
$ SELECT 632IDP00/QUIK/SOURCE/JLM/ASTERISK
$ LINK DATA,JLM
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/DATA
$ SELECT 632IDP00/QUIK/SOURCE/DATA/DATA
$ LINK DATADL
$ FORTY MAP,XREF,DECK
$ PRMFL X*,W,S,632IDP00/QUIK/OBJECT/DELETE
$ SELECT 632IDP00/QUIK/SOURCE/DATA/DELETE
$ LINK DATACH,DATADL
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/CHANGE
$ SELECT 632IDP00/QUIK/SOURCE/DATA/CHANGE
$ SELECT 632IDP00/QUIK/SOURCE/DATA/DESSCH
$ SELECT 632IDP00/QUIK/SOURCE/DATA/NXTDES

```

Figure 194. (Part 2 of 11)

```

$ SELECT 632IDP00/QUIK/SOURCE/DATA/VALPUT
$ LINK DATACR,DATACH
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/CREATE
$ SELECT 632IDP00/QUIK/SOURCE/DATA/CREATE
$ SELECT 632IDP00/QUIK/SOURCE/DATA/VALPUT
$ LINK DBMOD,DATA
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/DBMOD
$ SELECT 632IDP00/QUIK/SOURCE/DBMOD/DBMOD
$ SELECT 632IDP00/QUIK/SOURCE/DBMOD/DESTAB
$ LINK INDEXER,DBMOD
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/INDEXER
$ SELECT 632IDP00/QUIK/SOURCE/INDEXER/INDEXER
$ SELECT 632IDP00/QUIK/SOURCE/INDEXER/COMPLEX
$ SELECT 632IDP00/QUIK/SOURCE/INDEXER/CRTBLE
$ SELECT 632IDP00/QUIK/SOURCE/INDEXER/SETVAL
$ SELECT 632IDP00/QUIK/SOURCE/INDEXER/VLRADI
$ LINK PLANS,INDEXER
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/PLANSET
$ SELECT 632IDP00/QUIK/SOURCE/PLANSET/PLANSET
$ SELECT 632IDP00/QUIK/SOURCE/PLANSET/ADJUSTGP
$ SELECT 632IDP00/QUIK/SOURCE/PLANSET/CALCOMP
$ SELECT 632IDP00/QUIK/SOURCE/PLANSET/GRPEM
$ SELECT 632IDP00/QUIK/SOURCE/PLANSET/PRINTGP
$ SELECT 632IDP00/QUIK/SOURCE/PLANSET/SRTTGT
$ SELECT 632IDP00/QUIK/SOURCE/PLANSET/TANKER
$ SELECT 632IDP00/QUIK/SOURCE/PLANSET/VLRADP
$ LINK PREP,PLANS
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/PREPALOC
$ SELECT 632IDP00/QUIK/SOURCE/PREPALOC/PREPALOC
$ SELECT 632IDP00/QUIK/SOURCE/PREPALOC/DEPROUT
$ SELECT 632IDP00/QUIK/SOURCE/PREPALOC/FACTORCG
$ SELECT 632IDP00/QUIK/SOURCE/PREPALOC/FIXWEP
$ SELECT 632IDP00/QUIK/SOURCE/PREPALOC/MAKECHG
$ SELECT 632IDP00/QUIK/SOURCE/PREPALOC/PENROUT
$ SELECT 632IDP00/QUIK/SCURCE/PREPALOC/WEPPREP
$ SELECT 632IDP00/QUIK/SOURCE/PREPALOC/TGTPREP
$ LINK EDIT,PREP
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/EDITDB
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/EDITDB
$ LINK ECOUNT
$ FORTY MAP,XREF,DECK

```

Figure 194. (Part 3 of 11)

```

$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/COUNTS
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/COUNTS
$ LINK EGENED,ECOUNT
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/GENEDIT
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/GENEDIT
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/BUILDTAB
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/FORMLOC
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/SETFLD
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/SWTH
$ LINK ENORMA,EGENED
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/NORMAL
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/NORMAL
$ LINK EPROCE,ENORMA
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/PROCEDIT
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/PROCEDIT
$ SELECT 632IDP00/QUIK/SOURCE/EDITDB/XWITH
$ LINK REPORT,EDIT
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/REPORT
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/REPORT
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/DSPPUT
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/TABMNT
$ LINK RPTDSN
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/DESIGN
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/DESIGN
$ LINK RPTALT,RPTDSN
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/ALTER
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/ALTER
$ LINK RPTDMK,RPTALT
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/DSPMAK
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/DSPMAK
$ LINK RPTPRN,RPTDMK
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/PRINT
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/PRINT
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/PRNATD
$ SELECT 632IDP00/QUIK/SOURCE/REPORT/XDEFN
$ LINK SRM,REPORT
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/SRM

```

Figure 194. (Part 4 of 11)

```

$ SELECT 632IDP00/QUIK/SOURCE/SRM/SRM
$ LINK EIM,SRM
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/EIM
$ SELECT 632IDP00/QUIK/SOURCE/EIM/EIM
$ SELECT 632IDP00/QUIK/SOURCE/EIM/CONVLL
$ LINK BSIDAC
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/SIDAC
$ SELECT 632IDP00/QUIK/SOURCE/EIM/SIDAC
$ LINK BOTHER,BSIDAC
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/BLDOTH
$ SELECT 632IDP00/QUIK/SOURCE/EIM/BLDOTH
$ SELECT 632IDP00/QUIK/SOURCE/EIM/XEDEFN
$ LINK BTABLE,BOTHER
$ FORTY MAP,XREF,DECK
$ PRMFL 632IDP00/QUIK/OBJECT/TABLE
$ SELECT 632IDP00/QUIK/SOURCE/EIM/TABLE
$ LINK PLOTTT,BTABLE
$ FORTY MAP,XREF,DECK
$ DRMFL 632IDP00/QUIK/OBJECT/PLOTDATA
$ SELECT 632IDP00/QUIK/SOURCE/EIM/PLOTDATA
$ SELECT 632IDP00/QUIK/SOURCE/EIM/PICS
$ SELECT 632IDP00/QUIK/SOURCE/EIM/PROJCT
$ LINK PLOTIT,PLOTTT
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/PLOTIT
$ SELECT 631IDP00/QUIK/SOURCE/EIM/PLOTIT
$ SELECT 631IDP00/QUIK/SOURCE/EIM/FNDSRT
$ SELECT 631IDP00/QUIK/SOURCE/EIM/INTRPL
$ SELECT 631IDP00/QUIK/SOURCE/EIM/PICS
$ SELECT 631IDP00/QUIK/SOURCE/EIM/PBLLOFF
$ SELECT 631IDP00/QUIK/SOURCE/EIM/PBLLOFF
$ SELECT 631IDP00/QUIK/SOURCE/EIM/PLOTINIT
$ SELECT 631IDP00/QUIK/SOURCE/EIM/PROJCT
$ SELECT 631IDP00/QUIK/SOURCE/EIM/SUBPLOT
$ SELECT 631IDP00/QUIK/SOURCE/EIM/SUBREAD
$ LINK ALOC,EIM
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/ALOC
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/ALOC
$ LINK ALCINT
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/ALCINT
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/INITAL

```

Figure 194. (Part 5 of 11)

```

$ SELECT 632IDP00/QUIK/SOURCE/ALOC/CNCLST
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/DATGRP
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/FLOCRS
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/MRVRST
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/PRNPUT
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/RDMUL
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/RDPRNZ
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/RDSET
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/RDSMAT
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/RNGALT
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/SETABLE
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/TIMEPRT
$ LINK ALCMUL,ALCINT
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/ALCMUL
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/MULCON
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/ADDSAL
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/ASGOUT
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/BOMPRM
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/MYAPOS
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/PRNTALL
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/PRNTCON
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/PRNTNOW
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/TABLEMUP
$ LINK FGD
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/FGD
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/FRSTGD
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/CRDCAL
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/FLGCHK
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/INICRD
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/NXSPLT
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/PKCALC
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/PRNTOF
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/RECON
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/SETPAY
$ LINK SGD,FGD
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/SGD
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/SCNDGD
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/NXSPLT
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/RECON
$ SELECT 632IDP00/QUIK/SOURCE/ALOC/SETPAY
$ LINK STAL,SGD
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/STAL

```

Figure 194. (Part 6 of 11)

```

$      SELECT 632IDP00/QUIK/SOURCE/ALOC/STALL
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/FORMATS
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/KMUP
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/LAMGET
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/PREMIUMS
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/PRNTOS
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/SALVAL
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/SPLIT
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/WAD
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/WADOUT
$      LINK    DEFAL, STAL
$      FORTY   MAP, XREF, DECK
$      PRMFL   C*, W, S, 632IDP00/QUIK/OBJECT/DEFAL
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/DEFALOC
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/FMUP
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/LAMGET
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/PREMIUMS
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/PRNTOD
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/RESVAL
$      SELECT 632IDP00/QUIK/SOURCE/ALOC/SALVAL
$      LINK    EVAL, ALOC
$      FORTY   MAP, XREF, DECK
$      PRMFL   C*, W, S, 632IDP00/QUIK/OBJECT/EVALALOC
$      SELECT 632IDP00/QUIK/SOURCE/EVALALOC/EVALALOC
$      SELECT 632IDP00/QUIK/SOURCE/EVALALOC/EVALPLAN
$      SELECT 632IDP00/QUIK/SOURCE/EVALALOC/EVAL2
$      SELECT 632IDP00/QUIK/SOURCE/EVALALOC/PREVAL
$      SELECT 632IDP00/QUIK/SOURCE/EVALALOC/SSSPCALC
$      SELECT 632IDP00/QUIK/SOURCE/EVALALOC/TGTMODIF
$      SELECT 632IDP00/QUIK/SOURCE/EVALALOC/WPNMODIF
$      LINK    DGZSEL, EVAL
$      FORTY   MAP, XREF, DECK
$      PRMFL   C*, W, S, 632IDP00/QUIK/OBJECT/ALOCOUT
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/ALOCOUT
$      LINK    OFFSET
$      FORTY   MAP, XREF, DECK
$      PRMFL   C*, W, S, 632IDP00/QUIK/OBJECT/OFFSET
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/COMPRESS
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/CUMINV
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/DGZ
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/ERGOT1
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/FINDMIN
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/FINDMIN
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/F2BMIN
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/GRADF
$      SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/MOVE

```

Figure 194. (Part 7 of 11)

```

$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/PERTBLD
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/PROCCOMP
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/SEECALC
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/SEEINPUT
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/VAL
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/VMARG
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/WEPGET
$ LINK ASGSET,OFFSET
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/ASGSET
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/SUMPRN
$ LINK MINIO,ASGSET
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/MINIO
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/MINIOUT
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/CONVLL
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/FINDTIME
$ SELECT 632IDP00/QUIK/SOURCE/ALOCOUT/INFORN
$ LINK POST,DGZSEL
$ FORTY MAP,XREF,DECK
$ PRMFL C*,W,S,632IDP00/QUIK/OBJECT/POSTALOC
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/POSTALOC
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/CENTROID
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/CHGPLAN
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/CORRPARM
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/DIFF
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/DUMPSRT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/EVALB
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/EVALOA
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/EVALOB
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/FLTPLAN
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/FLTRROUTE
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/GENRAID
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/GETGROUP
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/GETSORT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/INITOPT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/INPOTGT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/NEXTFLT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/NOCORR
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/OPTRAID
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/OUTPOTGT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/OUTSRT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/PRERAID
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/PRINTIT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/PRNTF
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/SETFLAG

```

Figure 194. (Part 8 of 11)

```

$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/SORTOPT
$ SELECT 632IDP00/QUIK/SOURCE/POSTALOC/TGTASGN
$ LINK FOOT, POST
$ FORTY MAP, XREF, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/FOOTPRNT
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/FOOTPRNT
$ LINK OPTS
$ FORTY MAP, XRED, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/FOOTOPTS
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/TAB2INPT
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/MKAOS
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/PRAOS
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/PRINSETS
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/TAOS
$ LINK SETS, OPTS
$ FORTY MAP, XREF, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/FOOTSETS
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/NEWCOOR
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/SETDATA
$ LINK PLAN, SETS
$ FORTY MAP, XREF, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/FOOTPLAN
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/AXES
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/DRIVER
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/ELLIPSE
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/PATHFIND
$ SELECT 632IDP00/QUIK/SOURCE/FOOTPRNT/XAOS
$ LINK ASGN, PLAN
$ FORTY MAP, XREF, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/FOOTASGN
$ LINK PLANO, ROOT
$ FORTY MAP, XREF, DECK
$ PRMFL X*, W, S, 632IDP00/QUIK/OBJECT/PLANO
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/PLANOUT
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/CLINDATA
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/CONVLL
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/GEOGET
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/SNAPCON
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/SEPDATA
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/XLL
$ LINK PLNT
$ FORTY MAP, XREF, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/PLNT
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/PLNTPLAN
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/ALTPLAN
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/ADJUST

```

Figure 194. (Part 9 of 11)

```

$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/ALTERR
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/CHGTIM
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/DECOYADD
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/DISTIME
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/FINDME
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/FLTSORT
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/FLYPOINT
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/INITANK
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/KERPLUNK
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/LAUNCH
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/LNCHDATA
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/PLAN
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/PLANBOMB
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/PLANTMIS
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/POST
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/POSTLAUN
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/SNAPIT
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/SNAPOUT
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/SORBOMB
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/SWCHALT
$ LINK INTR, PLNT
$ FORTY MAP, XREF, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/INTR
$ SELECT 631IDP00/QUIK/SOURCE/PLANOUT/INTRFACE
$ SELECT 631IDP00/QUIK/SOURCE/PLANOUT/ABOUT
$ SELECT 631IDP00/QUIK/SOURCE/PLANOUT/FINDTIME
$ SELECT 631IDP00/QUIK/SOURCE/PLANOUT/IAZIM
$ SELECT 631IDP00/QUIK/SOURCE/PLANOUT/IFSET
$ SELECT 631IDP00/QUIK/SOURCE/PLANOUT/IFUNCT
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/INFORM
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/NOP
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/PRNTOFFS
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/RDCLAUSE
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/STOUT
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/TYPFIND
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/XSET
$ LINK TANK, INTR
$ FORTY MAP, XREF, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/TANK
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/PLANTANK
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/PRNTAB
$ SELECT 632IDP00/QUIK/SOURCE/PLANOUT/VAM
$ LINK DMAKE, PLANO
$ FORTY MAD, XRED, DECK
$ PRMFL C*, W, S, 632IDP00/QUIK/OBJECT/DATAMAKE
$ SELECT 632IDP00/QUIK/SOURCE/DATAMAKE/DATAMAKE

```

Figure 194. (Part 10 of 11)

```

$ SELECT 632IDP00/QUIK/SOURCE/DATAMAKE/CALCOMP
$ SELECT 632IDP00/QUIK/SOURCE/DATAMAKE/COMPLEX
$ SELECT 632IDP00/QUIK/SOURCE/DATAMAKE/CRTABLE
$ SELECT 632IDP00/QUIK/SOURCE/DATAMAKE/SRTTGT
$ SELECT 632IDP00/QUIK/SOURCE/DATAMAKE/VLRADI
$ SELECT 632IDP00/QUIK/SOURCE/DATAMAKE/VLRADP
$ EXECUTE DUMP,DEBUG,NREST,JREST
$ LIMITS 30,60K,-3K,30K
$ FFILE P*,LGU/(06,42,43,11,12)
$ PRMFL H*R/W,R,632IDP00/QUIK/COP/HSTAR
$ PRMFL QD,R/W,R,632IDP00/QUIK/COP/IDS
$ PRMFL UL,R/W,R,632IDP00/QUIK/LIBRARY/UTIL
$ PRMFL PL,R,S,LIBRARY/PLOTLIB
$ FILE 02,X2R,100L
$ FILE 08,X8R,100L
$ FILE 19,X19R,100L
$ TAPE9 20,X20D,,12345,,INPUT-JAD
$ FILE 21,X21R,50L
$ FFILE 21,NBUFFS/2
$ FILE 22,X22R,50L
$ FFILE 22,NBUFFS/2
$ FILE 23,X23R,50L
$ FILE 23,NBUFFS/2
$ FILE 24,X24R,50L
$ FFILE 24,NBUFFS/2
$ FILE 25,X25R,100R
$ FILE 30,X30R,10L
$ TAPE9 31,X31D,,54321,,OUTPUT-SPILLTAPE
$ TAPE9 32,X32D,,54345,,SAVE-RESTORE-TAPE
$ TAPE9 35,X35D,,67890,,OUTPUT-TAPE-1
$ TAPE0 36,X36D,,98765,,OUTPUT-TAPE-2
$ DATA I*
$ ENDJOB

```

***EOF

Figure 194. (Part 11 of 11)

```

$ IDENT 5162,COMPL,314,I M A PROG,631,12
$ USERID 632IDP99$PASSWORD/UZZ
$ LIMITS 30,60K,,10K
$ OPTION FORTRAN,NOGO
$ LIBRARY UL,PL
$ LOWLOAD
$ ENTRY .....
$ SELECT 632IDP00/QUIK/OBJECT/COP
$ SELECT 632IDP00/QUIK/OBJECT/QDATA
$ USE .QMAX/1/,.QAREA/3126/,.QMIN/1/,.FRRD.
$ LINK BOOTT
$ SELECT 632IDP00/QUIK/OBJECT/BOOT
$ LINK TABSTR,BOOT
$ USE TABLZ/808/
$ LINK ERRF
$ SELECT 632IDP00/QUIK/OBJECT/ERRFND
$ LINK INPT,ERRF
$ SELECT 632IDP00/QUIK/OBJECT/INPTRN
$ LINK JLM,TABSTR
$ SELECT 632IDP00/QUIK/OBJECT/JLM
$ LINK ASSI
$ SELECT 632IDP00/QUIK/OBJECT/ASSIGN
$ LINK SELE,ASSI
$ SELECT 632IDP00/QUIK/OBJECT/SELECT
$ LINK ASTE,SELE
$ SELECT 632IDP00/QUIK/OBJECT/ASTERISK
$ LINK DATA,JLM
$ SELECT 632IDP00/QUIK/OBJECT/DATA
$ LINK DATADL
$ SELECT 632IDP00/QUIK/OBJECT/DELETE
$ LINK DATACH,DATADL
$ SELECT 632IDP00/QUIK/OBJECT/CHANGE
$ LINK DATACR,DATACH
$ SELECT 632IDP00/QUIK/OBJECT/CREATE
$ LINK DBMOD,DATA
$ SELECT 632IDP00/QUIK/OBJECT/DBMOD
$ LINK INDXER,DBMOD
$ SELECT 632IDP00/QUIK/OBJECT/INDEXER
$ LINK PLANS,INDXER
$ SELECT 632IDP00/QUIK/OBJECT/PLANSET
$ LINK PREP,PLANS
$ SELECT 632IDP00/QUIK/OBJECT/PREPALOC
$ LINK EDIT,PREP
$ SELECT 632IDP00/QUIK/OBJECT/EDITDB
$ LINK ECOUNT
$ SELECT 632IDP00/QUIK/OBJECT/COUNTS

```

Figure 195. H* Creation From Object (Part 1 of 4)

```

$ LINK EGENED, ECOUNT
$ SELECT 632IDP00/QUIK/OBJECT/GENEDIT
$ LINK ENORMA, EGENED
$ SELECT 632IDP00/QUIK/OBJECT/NORMAL
$ LINK EPROCE, ENORMA
$ SELECT 632IDP00/QUIK/OBJECT/PROCEDIT
$ LINK REPORT, EDIT
$ SELECT 632IDP00/QUIK/OBJECT/REPORT
$ LINK RPTDSN
$ SELECT 632IDP00/QUIK/OBJECT/DESIGN
$ LINK RPTALT, RPTDSN
$ SELECT 632IDP00/QUIK/OBJECT/ALTER
$ LINK RPTDMK, RPTALT
$ SELECT 632IDP00/QUIK/OBJECT/DSPMAK
$ LINK REPPRN, RPTDMK
$ SELECT 632IDP00/QUIK/OBJECT/PRINT
$ LINK SRM, REPORT
$ SELECT 632IDP00/QUIK/OBJECT/SRM
$ LINK EIM, SRM
$ SELECT 632IDP00/QUIK/OBJECT/EIM
$ LINK BSIDAC
$ SELECT 632IDP00/QUIK/OBJECT/SIDAC
$ LINK BOTHER, BSIDAC
$ SELECT 632IDP00/QUIK/OBJECT/BLDOTH
$ LINK BTABLE, BOTHER
$ SELECT 632IDP00/QUIK/OBJECT/TABLE
$ LINK PLOTTT, BTABLE
$ SELECT 632IDP00/QUIK/OBJECT/PLOTDATA
$ LINK PLOTIT, PLOTTT
$ SELECT 632IDP00/QUIK/OBJECT/PLOTIT
$ LINK ALOC, EIM
$ SELECT 632IDP00/QUIK/OBJECT/ALOC
$ LINK ALCINT
$ SELECT 632IDP00/QUIK/OBJECT/ALCINT
$ LINK ALCMUL, ALCINT
$ SELECT 632IDP00/QUIK/OBJECT/ALCMUL
$ LINK FGD
$ SELECT 632IDP00/QUIK/OBJECT/FGD
$ LINK SGD, FGD
$ SELECT 632IDP00/QUIK/OBJECT/SGD
$ LINK STAL, SGD
$ SELECT 632IDP00/QUIK/OBJECT/STAL
$ LINK DEFAL, STAL
$ SELECT 632IDP00/QUIK/OBJECT/DEFAL
$ LINK EVAL, ALOC
$ SELECT 632IDP00/QUIK/OBJECT/EVALALOC

```

Figure 195. (Part 2 of 4)

```

$ LINK DGZSEL,EVAL
$ SELECT 632IDP00/QUIK/OBJECT/ALOCOUT
$ LINK OFFSET
$ SELECT 632IDP00/QUIK/OBJECT/OFFSET
$ LINK ASGSET,OFFSET
$ SELECT 632IDP00/QUIK/OBJECT/ASGSET
$ LINK MINIO,ASGSET
$ SELECT 632IDP00/QUIK/OBJECT/MINIO
$ LINK POST,DGZSEL
$ SELECT 632IDP00/QUIK/OBJECT/POSTALOC
$ LINK FOOT,POST
$ SELECT 632IDP00/QUIK/OBJECT/FOOTPRNT
$ LINK OPTS
$ SELECT 632IDP00/QUIK/OBJECT/FOOTOPTS
$ LINK SETS,OPTS
$ SELECT 632IDP00/QUIK/OBJECT/FOOTSETS
$ LINK PLAN,SETS
$ SELECT 632IDP00/QUIK/OBJECT/FOOTPLAN
$ LINK ASGN,PLAN
$ SELECT 632IDP00/QUIK/OBJECT/FOOTASGN
$ LINK PLANO,FOOT
$ SELECT 632IDP00/QUIK/OBJECT/PLANO
$ LINK PLNT
$ SELECT 632IDP00/QUIK/OBJECT/FLNT
$ LINK INTR,PLNT
$ SELECT 632IDP00/QUIK/OBJECT/INTR
$ LINK TANK,INTR
$ SELECT 632IDP00/QUIK/OBJECT/TANK
$ LINK DMAKE,PLANO
$ SELECT 632IDP00/QUIK/OBJECT/DATMAKE
$ EXECUTE DUMP,DEBUG,NREST,JREST
$ LIMITS 10,60K,-3K,10K
$ FFILE PX,LGU/(06,42,43,11,12)
$ PRMFL H*,R/W,R,632IDP00/QUIK/COP/HSTAR
$ PRMFL QD,R/W,R,631IDP00/QUIK/COP/IDS
$ PRMFL UL,R/W,R,631IDP00/QUIK/LIBRARY/UTIL
$ PRMFL PL,R,S,LIBRARY/PLOTLIB
$ FILE 02,X2R,100L
$ FILE 08,X8R,100L
$ FILE 19,X19R,100L
$ TAPE9 20,X20D,,12345,,INPUT-JAD
$ FILE 21,X21R,50L
$ FFILE 21,NBUFFS/2
$ FILE 22,X22R,50L
$ FFILE 22,NBUFFS/2
$ FILE 23,X23R,50L

```

Figure 195. (Part 3 of 4)

```

$      FFILE  23,NBUFFS/2
$      FILE   24,X24R,50L
$      FFILE  24,NBUFFS/2
$      FILE   25,X25R,100R
$      FILE   30,X30R,10L
$      TAPE9  31,X31D,,54321,,OUTPUT-SPILLTAPE
$      TAPE9  32,X32D,,54345,,SAVE-RESTORE-TAPE
$      TAPE9  35,X35D,,67890,,OUTPUT-TAPE-1
$      TAPE9  36,X36D,,98765,,OUTPUT-TAPE-2
$      DATA  I*
$      ENDJOB
***EOF

```

Figure 195. (Part 4 of 4)

```

$ IDENT 5162,CUTIL,314,I M A PROG,631,12
$ USERID 632IDP99$PASSWORD/UZZ
$ LIMITS 10,32K,,60K
$ FORTY DECK,MAP,XREF
$ FILE C*,X15,100L
$ SELECT 632IDP00/QUIK/MSRC/UTIL/ABORT
$ SELECT 632IDP00/QUIK/MSRC/UTIL/ACOS
$ SELECT 632IDP00/QUIK/MSRC/UTIL/ASIN
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/ATFNDR
$ SELECT 632IDP00/QUIK/MSRC/UTIL/ATN2PI
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/CINSGET
$ SELECT 632IDP00/QUIK/MSRC/UTIL/COT
$ SELECT 632IDP00/QUIK/MSRC/UTIL/DELLONG
$ SELECT 632IDP00/QUIK/MSRC/UTIL/DIFFLONG
$ SELECT 632IDP00/QUIK/MSRC/UTIL/DISTF
$ SELECT 632IDP00/QUIK/MSRC/UTIL/DOTLINE
$ SELECT 632IDP00/QUIK/SOURCE/FINDCLAS
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/FINDSIDE
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/FORMAK
$ GMAP DECK
$ FILE C*,X1S
$ SELECT 632IDP00/QUIK/MSRC/UTIL/GETCLOCK
$ FORTY DECK,MAP,XREF
$ FILE C*,X1S
$ SELECT 632IDP00/QUIK/MSRC/UTIL/GETDATE
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/GETNXT
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/GETSTR
$ SELECT 632IDP00,QUIK/SOURCE/UTIL/GETTAR
$ SELECT 632IDP00/QUIK/MSRC/UTIL/GETVALU
$ SELECT 632IDP00/QUIK/MSRC/UTIL/GLOG
$ SELECT 632IDP00/QUIK/MSRC/UTIL/HOUSKEEP
$ SELECT 632IDP00/QUIK/MSRC/UTIL/IGET
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/IGETHOB
$ SELECT 632IDP00/QUIK/MSRC/UTIL/IHASH
$ SELECT 632IDP00/QUIK/MSRC/UTIL/IMAX
$ SELECT 632IDP00/QUIK/MSRC/UTIL/INTERP
$ SELECT 632IDP00/QUIK/MSRC/UTIL/INTERPGC
$ SELECT 632IDP00/QUIK/MSRC/UTIL/INTPIECE
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/IOFL
$ SELECT 632IDP00/QUIK/MSRC/UTIL/IPUT
$ SELECT 632IDP00/QUIK/MSRC/UTIL/ISOFF
$ SELECT 632IDP00/QUIK/MSRC/UTIL/ITLE
$ SELECT 632IDP00/QUIK/MSRC/UTIL/IWANT
$ SELECT 632IDP00/QUIK/MSRC/UTIL/KEYMAKE
$ SELECT 632IDP00/QUIK/MSRC/UTIL/LATBT
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/LINKUP

```

Figure 196. Utility Library Creation (Part 1 of 3)

```

$      GMAP      DECK
$      FILE      C*,X1S
$      SELECT    631IDP00/QUIK/MSRC/UTIL/LOCF
$      FORTY     DECK,MAP,XREF
$      FILE      C*,X1S
$      SELECT    632IDP00/QUIK/MSRC/UTIL/LOCREAD
$      SELECT    632IDP00/QUIK/MSRC/UTIL/LREORDER
$      SELECT    632IDP00/QUIK/MSRC/UTIL/MAPEDGE
$      SELECT    632IDP00/QUIK/MSRC/UTIL/NUMDEL
$      SELECT    632IDP00/QUIK/MSRC/UTIL/NUMGET
$      SELECT    632IDP00/QUIK/SOURCE/UTIL/OFVAL
$      SELECT    632IDP00/QUIK/MSRC/UTIL/ORDER
$      SELECT    632IDP00/QUIK/MSRC/UTIL/PIECEIT
$      SELECT    632IDP00/QUIK/MSRC/UTIL/PIECENUM
$      SELECT    632IDP00/QUIK/MSRC/UTIL/PLTTIC
$      SELECT    632IDP00/QUIK/SOURCE/UTIL/PRIMHD
$      SELECT    632IDP00/QUIK/SOURCE/UTIL/PSREC
$      SELECT    632IDP00/QUIK/MSRC/UTIL/RANF
$      SELECT    632IDP00/QUIK/MSRC/UTIL/RANFSET
$      SELECT    632IDP00/QUIK/MSRC/UTIL/REORDER
$      SELECT    632IDP00/QUIK/MSRC/UTIL/RLBRT
$      SELECT    632IDP00/QUIK/MSRC/UTIL/SETDEF
$      GMAP      DECK
$      FILE      C*,X1S
$      SELECT    632IDP00/QUIK/MSRC/UTIL/SETORD
$      FORTY     DECK,AMP,XREF
$      FILE      C*,X1S
$      Select    632IDP00/QUIK/SOURCE/UTIL/SETSCH
$      SELECT    632IDP00/QUIK/MSRC/UTIL/SLOG
$      SELECT    532IDP00/QUIK/SOURCE/UTIL/SORTIT
$      SELECT    632IDP00/QUIK/MSRC/UTIL/SSKPC
$      SELECT    632IDP00/QUIK/MSRC/UTIL/STD
$      GMAP      DECK
$      FILE      C*,X1S
$      SELECT    632IDP00/QUIK/SOURCE/UTIL/SVTP
$      FORTY     DECK,MAP,XREF
$      FILE      C*,X1S
$      SELECT    632IDP00/QUIK/MSRC/UTIL/TAN
$      SELECT    632IDP00/QUIK/MSRC/UTIL/TICMAKE
$      SELECT    632IDP00/QUIK/MSRC/UTIL/TIMEDAY
$      SELECT    632IDP00/QUIK/MSRC/UTIL/TIMEME
$      SELECT    632IDP00/QUIK/SOURCE/UTIL/TOFM
$      SELECT    632IDP00/QUIK/SOURCE/UTIL/UNCODE
$      SELECT    632IDP00/QUIK/SOURCE/UTIL/VALFND
$      SELECT    632IDP00/QUIK/MSRC/UTIL/VALTAR
$      SELECT    632IDP00/QUIK/MSRC/UTIL/XCOORD

```

Figure 196. (Part 2 of 3)

```

$ SELECT 632IDP00/QUIK/SOURCE/UTIL/XMATH
$ SELECT 632IDP00/QUIK/SOURCE/UTIL/XWHERE
$ SELECT 632IDP00/QUIK/MSRC/UTIL/ZTAN
$ UTILITY
$ LIMITS 10,32K,,60K
$ FUTIL BB,CC,RWD/BB/,COPY/1R/
$ FUTIL AA,CC,RWD/AA/,COPY/9 F/
$ FUTIL BB,CC,RWD/BB/,SKIP/1R/,COPY/1R/,RWD/CC/
$ FILE AA,X1S,100L
$ FILE CC,X4S,100L
$ DATA BB,,COPY
$ INCLUDE
$ ENEDIT
$ ENDCOPY
$ FILEDIT NOSOURCE,OBJECT,INITIALIZE
$ FILE R*,X2S,100L
$ FILE *C,X4R,100L
$ PROGRAM RANLIB
$ FILE R*,X2R,100L
$ PRMFL A4,R/W,R,632IDP00/QUIK/LIBRARY/UTIL
$ ENDJOB

```

Figure 196. (Part 3 of 3)

APPENDIX C

PERFORM PROGRAM

This appendix contains maintenance information for the PERFORM program. PERFORM is an on-line program designed to generate remote job entry jobs for the QUICK system.

C.1 Purpose

PERFORM is an on-line interactive program which creates a file of Job Control Language (JCL) according to user directions. This file of JCL may be set up to perform any combination of the following functions:

- o Run QUICK
- o Recompile a module of QUICK and recreate the QUICK H* file
- o Recompile and recreate the QUICK utility subroutine library

C.2 Input

PERFORM is an interactive system and, therefore, obtains part of its input from user responses from the terminal. PERFORM also has three files (or sets of files) which it uses to build the job stream JCL.

- o A set of files each of which contains the JCL required to define a module and its linkage from object files. These files are all under the catalogue 631IDP00/TEST/COP/CANOF
- o A set of files each of which contains the JCL required to define a module and its linkage from source files. These files are all under this catalogue 631IDP00/TEST/COP/NEWCANOF
- o A file (631IDP00/PERFORM/VRBLIM) which details the various required limits for the system and which contains one record for each legal verb:
 - Column 1-8 Verb Name
 - Column 9-11 Maximum CPU Time
 - Column 12-14 Maximum Core Requirements
 - Column 15-17 Maximum Lines of Output
- o A file (631IDP00/PERFORM/IDENT2) which details the currently recognized users of the QUICK system. It also contains information concerning the data files used by said users. For each user, the file contains two records plus as many records as the number of data files used.

Record 1

Column 1-12	USERID
Column 13-24	Name used in PERFORM interactive output
Column 25	Number of files used
Column 26	User salutation parameter: 1 - Salutation suppressed 2 - Salutation activated

Record 2

Column 1-48	IDENT card for user
-------------	---------------------

Record 3 and following

Column 1-4	Source subcatalogue name (i.e., TEST, DUAL)
Column 5-8	Number of pages in data file
Column 9-32	Data file name

- o A file (631IDP00/PERFORM/SPFILE) which details an additional file whose description must be added to those normally found in the JCL. These descriptions are related to the verbs which the user has specified. Each additional file has one record.

Column 1-8	Verb
Column 9	Blank
Column 10-11	File code
Column 12	R for input file W for output file
Column 13-48	Description of file used in PERFORM output

C.4 Concept of Operation

PERFORM follows a series of steps at user direction. For details of the question and answer sequences see CSM UM 9-77, Volume I. Based on selection of 'MODE', PERFORM generates JCL for the job stream by adding a series of file names from the appropriate /CANOF and/or /NEWCANOF catalogues. For 'RUN' mode, only the current QUICK system H* is used. The user supplies a list of the verbs being used. From this list, PERFORM uses the information stored on file URBLIM to compute the LIMITS card parameters. The user may alter these. The user is then requested to add data files and/or lines of input. Then the SPFILE file is consulted to see if the user desires any special files. Finally, the LIMITS and other final cards are added and the user is instructed as to how to submit the job.

For 'COMPILE' mode, the QUICK H* is recreated. The files in the 'CANOF' catalogue which refer to normal modules in the QUICK system, except for those whose recompile is desired, are added to 'THEJOB' file. For those modules which are to be recompiled, the corresponding file in the

/NEWCANOF catalogue is added. The user is then asked if run mode is desired and if so, that mode is entered to build a job to execute the new system H*.

For 'OBJECT' mode, a temporary H* is created. This H* is built using the /CANOF files for the executive. Added to these are the /NEWCANOF files for modules the user wishes recompiled. Finally, any additional modules' /CANOF file is added at user request. The modules added and/or recompiled need not be normal elements of the system. After all desired modules have been added, run mode is entered to execute the temporary H*.

For 'INITIALIZE' mode, an activity initializing the IDA data file is added to THEJOB. The program then asks for mode again.

C.5 Major Subroutines

PERFORM has two subroutines. READIN scans user input and converts any lower case ASCII letters to upper case ASCII letters. IDENT builds the job stream IDENT card.

C.6 Common Blocks

PERFORM has only one common block LINE. This block contains the array LINE(80) which is used to communicate with subroutine READIN. Each word of LINE contains one character of input.

C.7 Program PERFORM

PURPOSE: To build JCL through on-line interaction

ENTRY POINTS:

FORMAL PARAMETERS: None

COMMON BLOCKS: LINE

SUBROUTINES CALLED: READIN, IDENT

CALLED BY: Entered through TSS subsystem YFORT

Method:

First the file THEJOB is attached and the IDENT card added to it. Next the mode is requested. If the reply is 'RUN', skip to statement 200 (figure 197). If the reply in 'COMPILE' the program asks for a list of the modules changed. A reply of 'HELP' will list the modules. A reply of 'UTILITY' or 'UTIL' causes '631IDP00/TEST/COP/CANOF/UTIL' to be added to THEJOB. Now the program goes through all modules adding files from the CANOF catalogue for unchanged modules and from the NEWCANOF catalogue for changed modules. Then the user is asked if run mode is desired. If not skip to statement 300 (figure 197).

If reply to 'MODE' is 'OBJECT' the program asks for modules to be recompiled and any other modules needed. These are added to the executive to build a temporary H*. Control then passes to the run mode. If reply to 'MODE' is 'INITIALIZE' an IDS initialization activity (Program QUTI) is added to 'THEJOB' and a new mode is requested.

Statement 200

CANOF/RUN is added to THEJOB. The VRBLIM file is attached, read in and detached. The user now inputs a list of the desired verbs. If the reply is 'HELP', the legal verbs are listed. The program now calculates the run limits. The limits are displayed and the user given the opportunity to change them. Now the user is asked to input data file names. Each file name is added to THEJOB. When the reply 'DONE' is encountered, the user is asked if additional data is desired. Otherwise the user is asked to input core images which are added to THEJOB. When a blank core is input the user is asked if additional files are desired and if so control returns to the input of data files.

When no more input is desired, the program reads file SPFILE for any special files that the input verbs may require. The user is asked to select the JCL format for any such special file.

Statement 300

The LIMITS card is added to THEJOB plus any final cards and the user is instructed as to how to execute the JCL. Finally, the CALLSS system routine is called and CARDIN is entered.

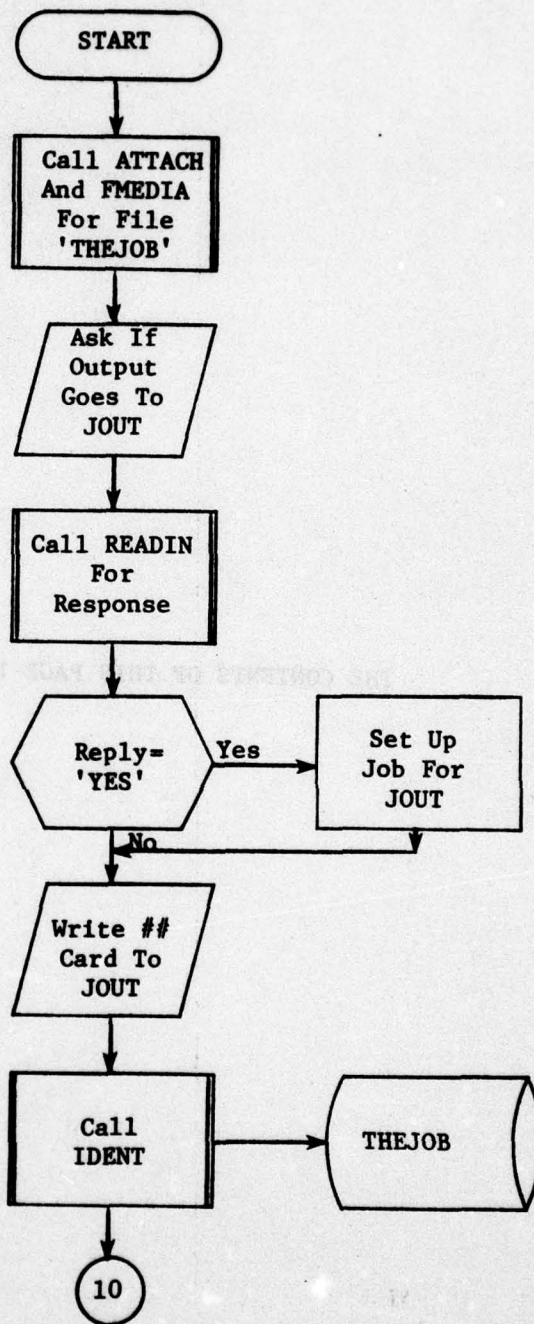


Figure 197. Program PERFORM (Part 1 of 14)

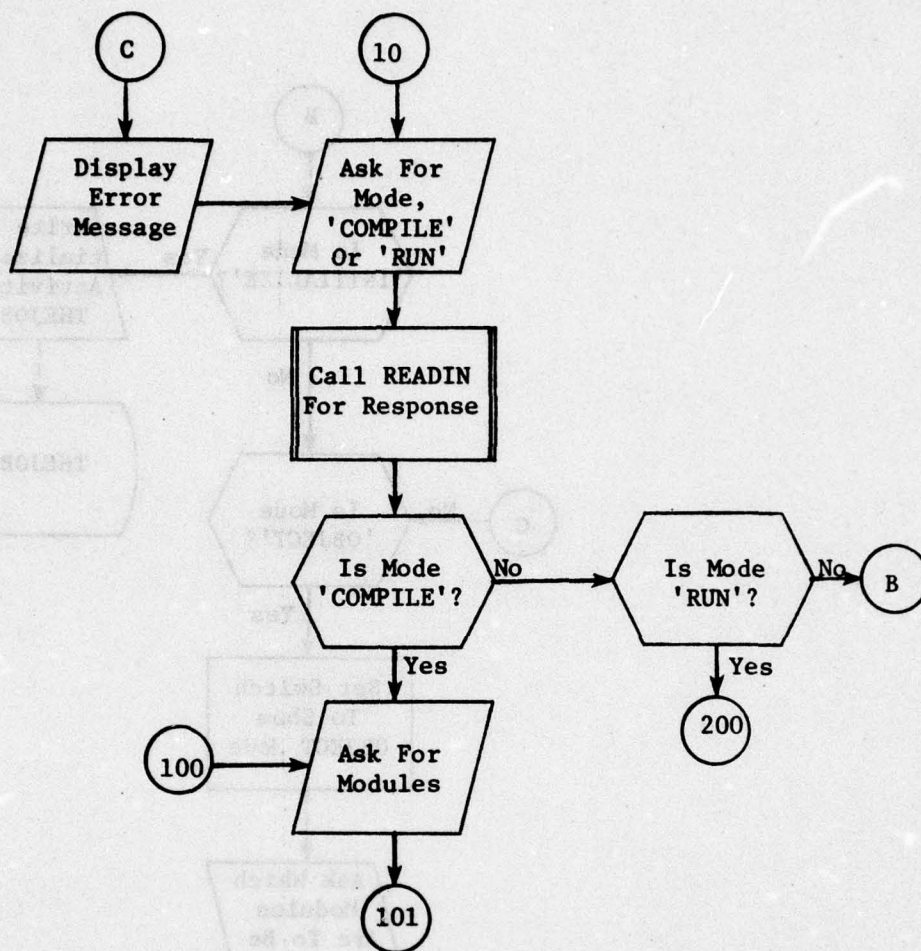


Figure 197. (Part 2 of 14)

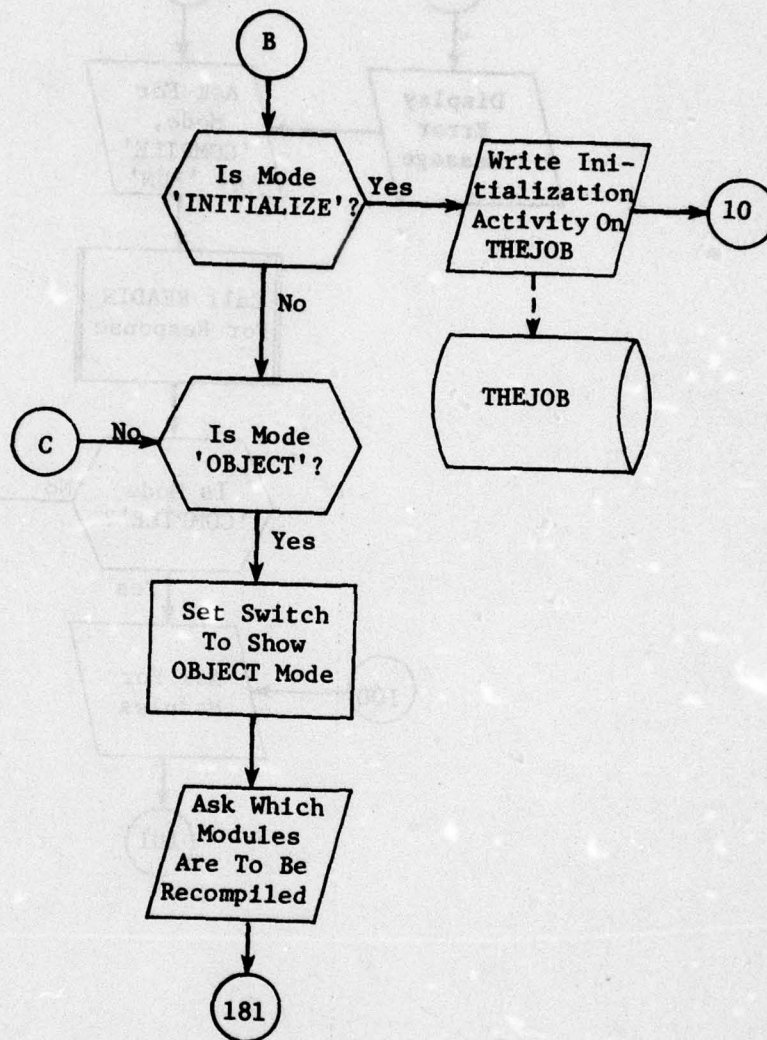


Figure 197. (Part 3 of 14)

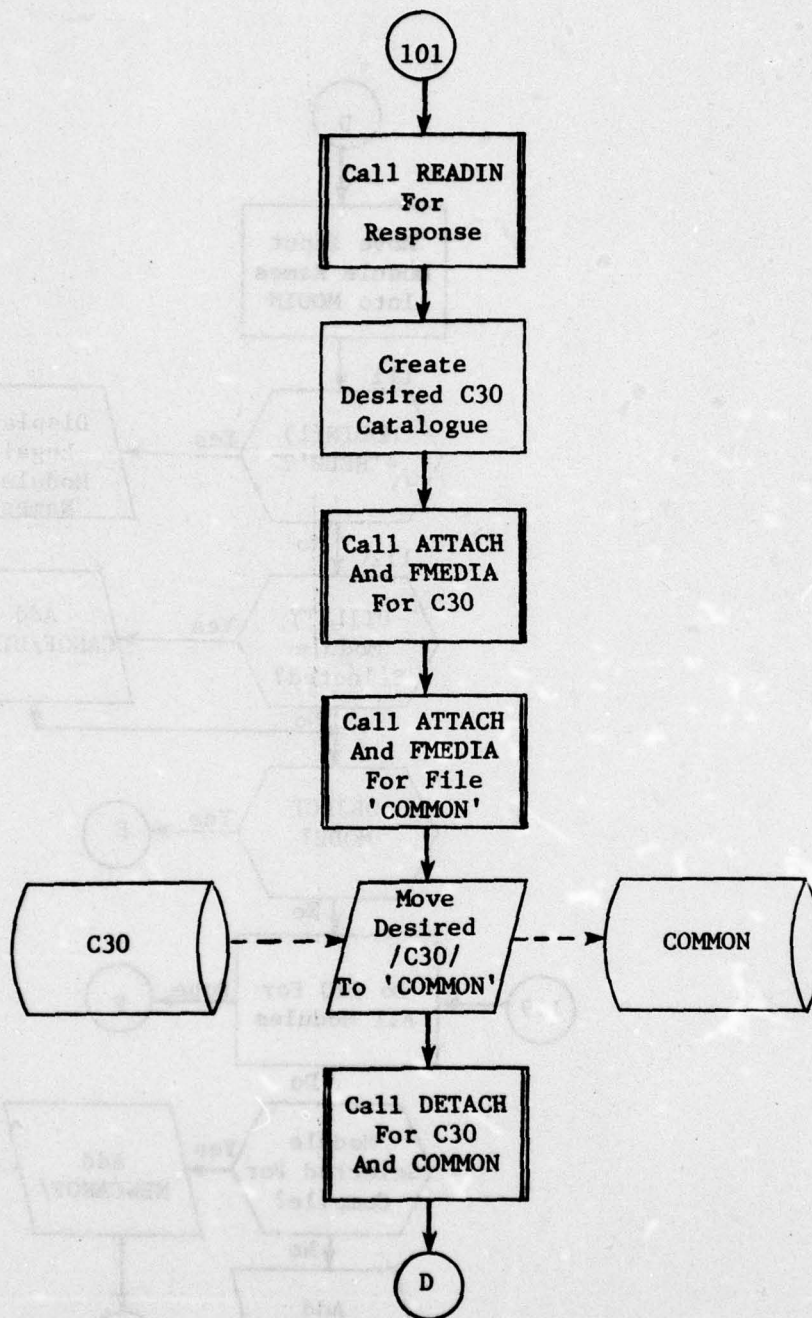


Figure 197. (Part 4 of 14)

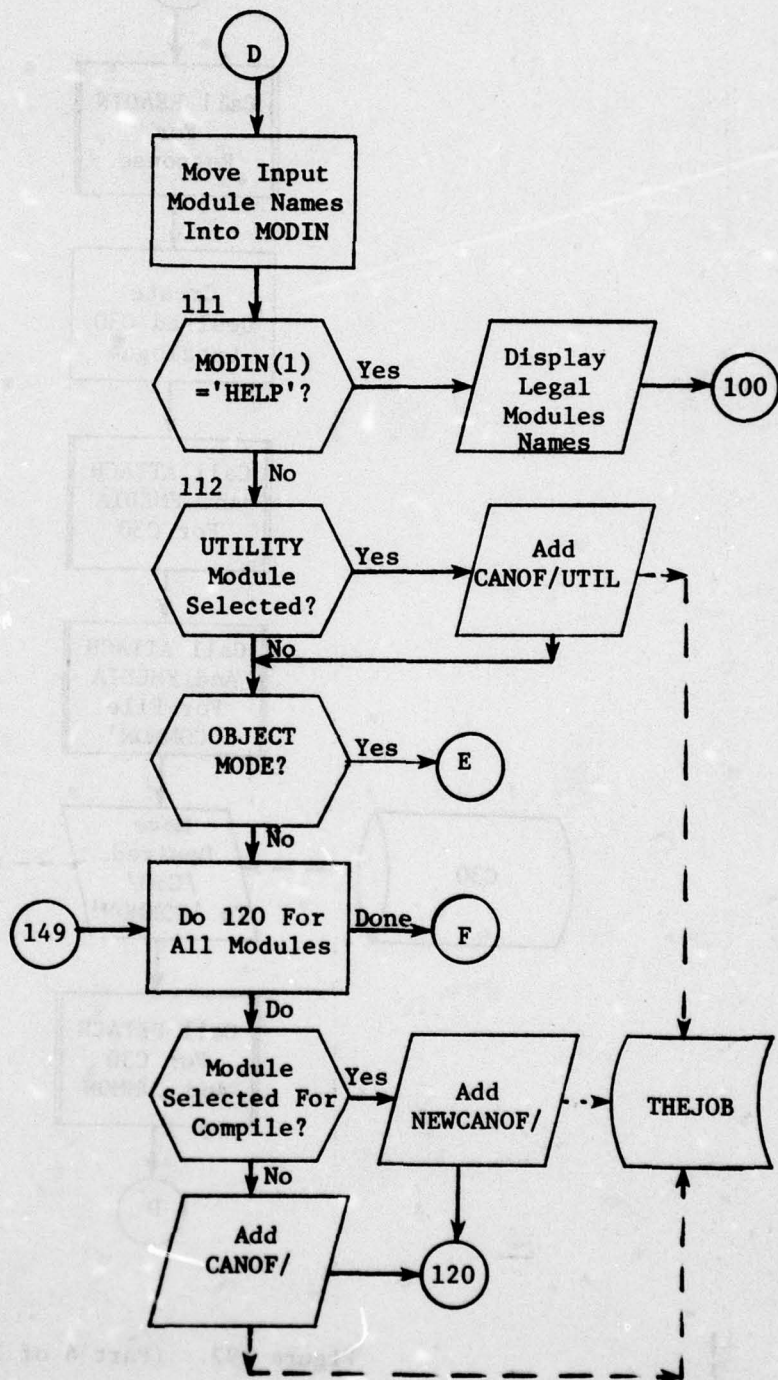


Figure 197. (Part 5 of 14)

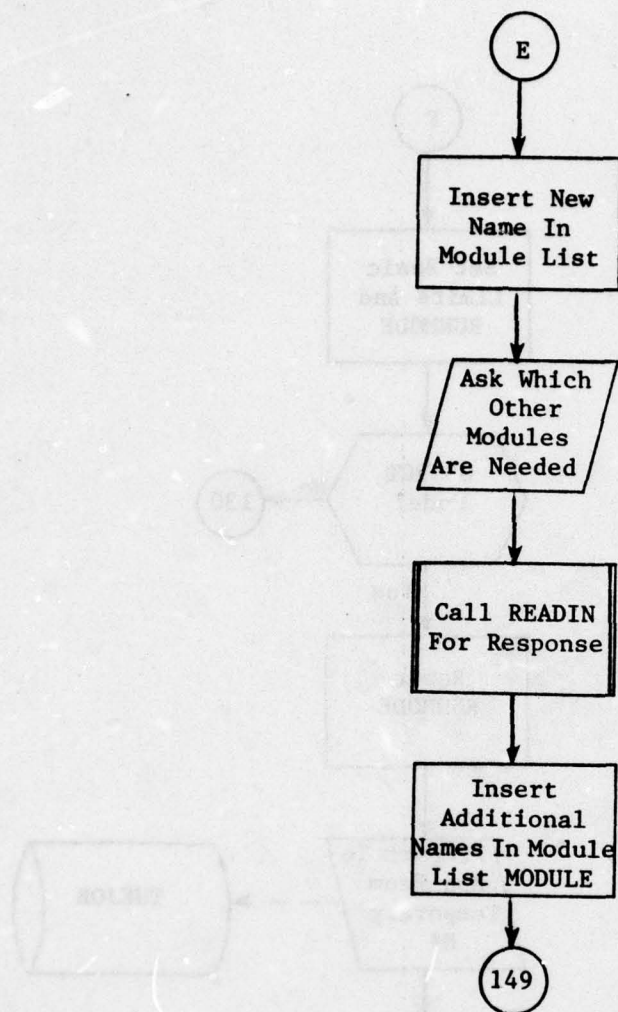


Figure 197. (Part 6 of 14)

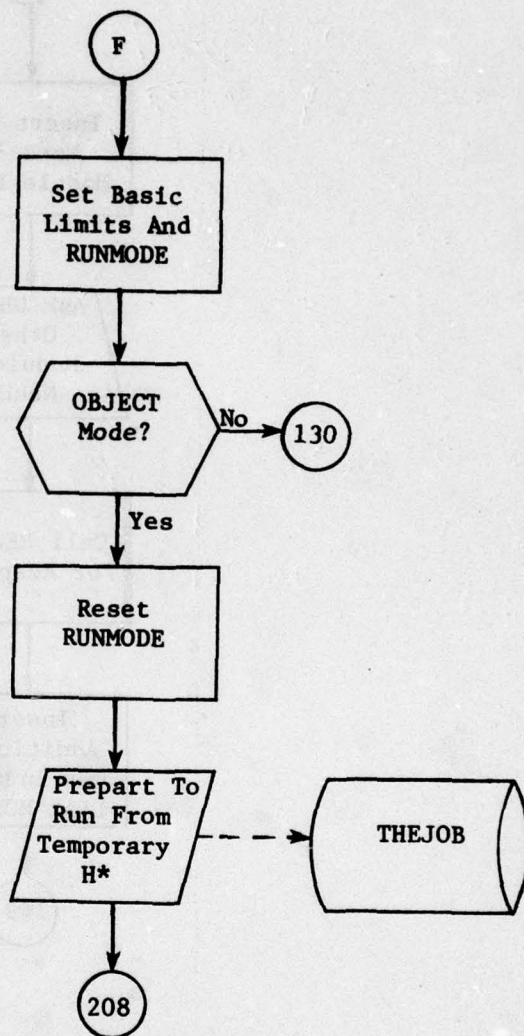


Figure 197. (Part 7 of 14)

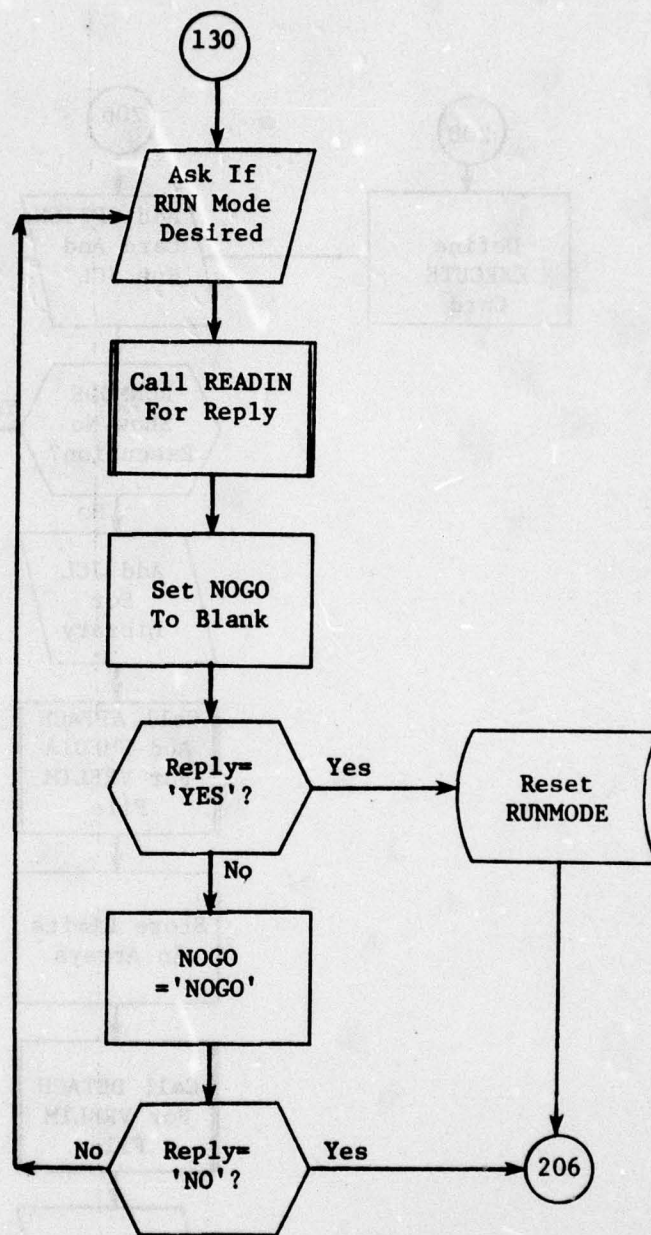


Figure 197. (Part 8 of 14)

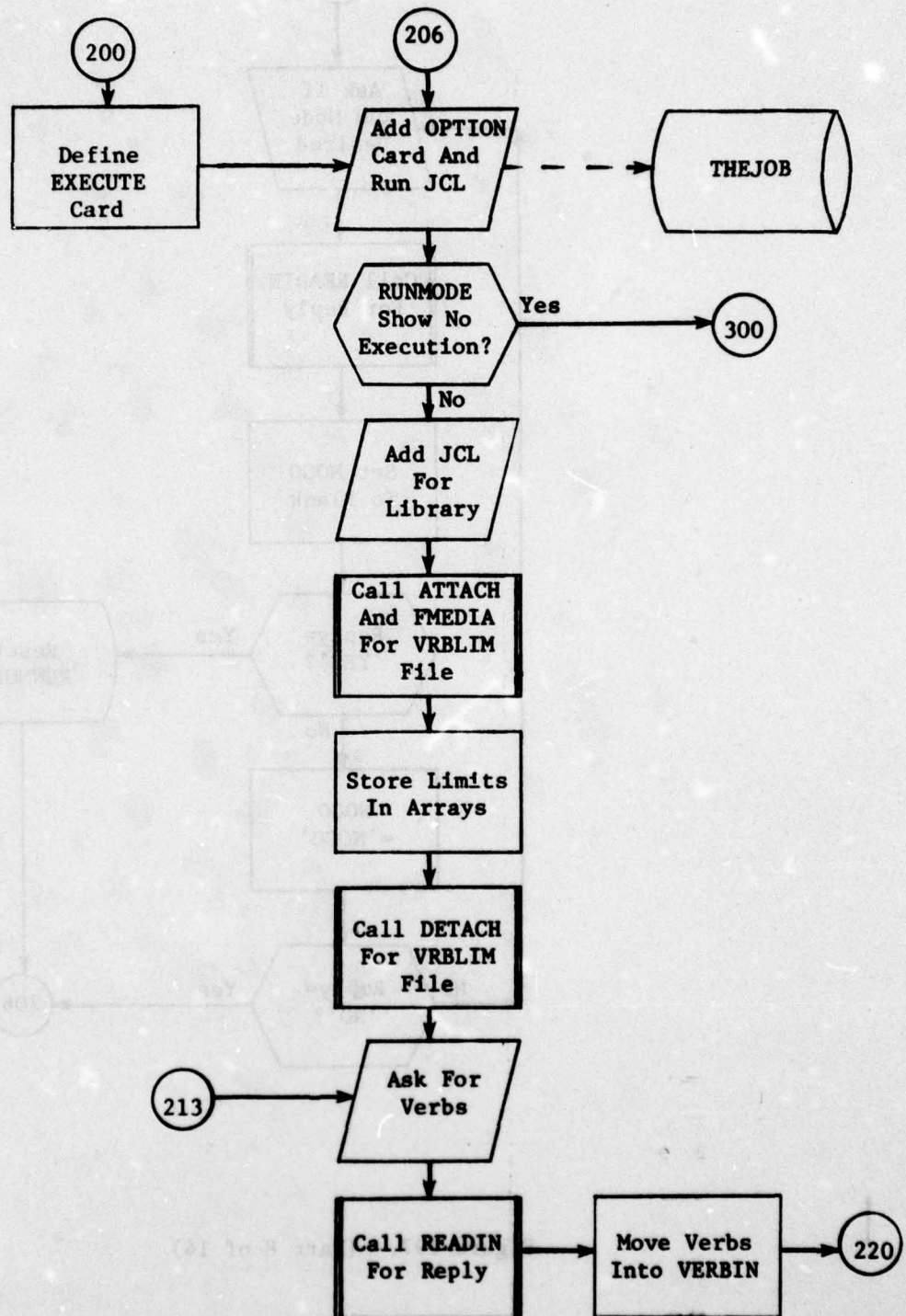


Figure 197. (Part 9 of 14)

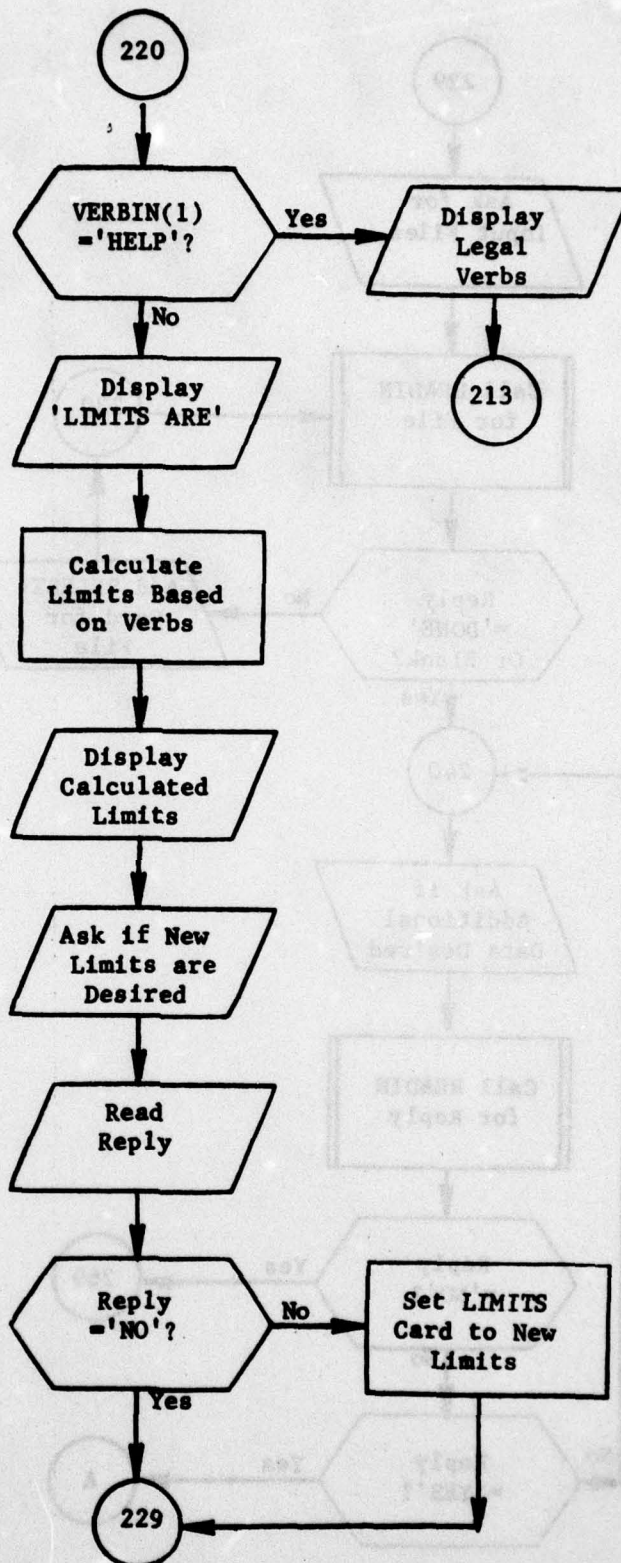


Figure 197. (Part 10 of 14)

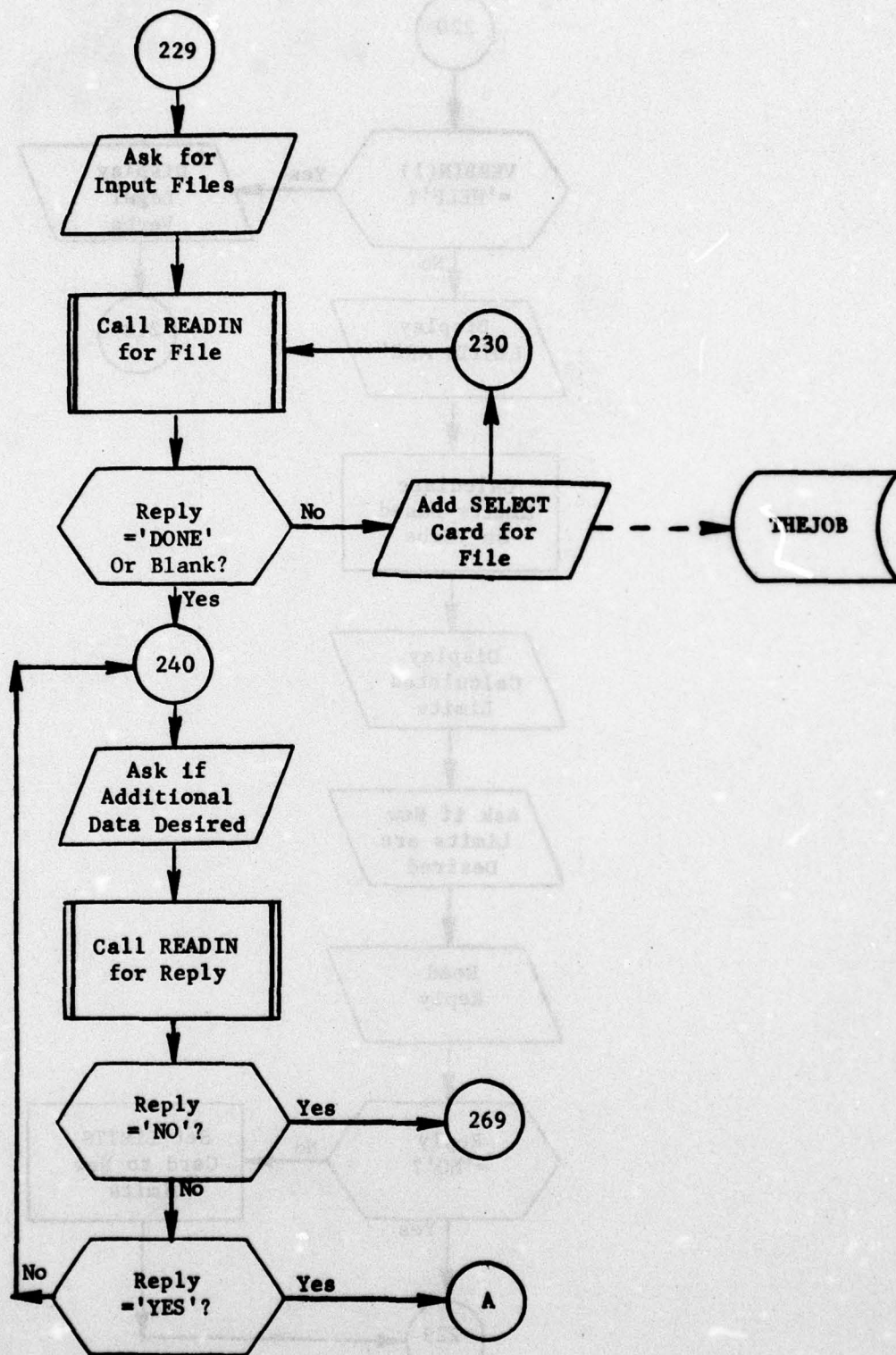


Figure 197. (Part 11 of 14)

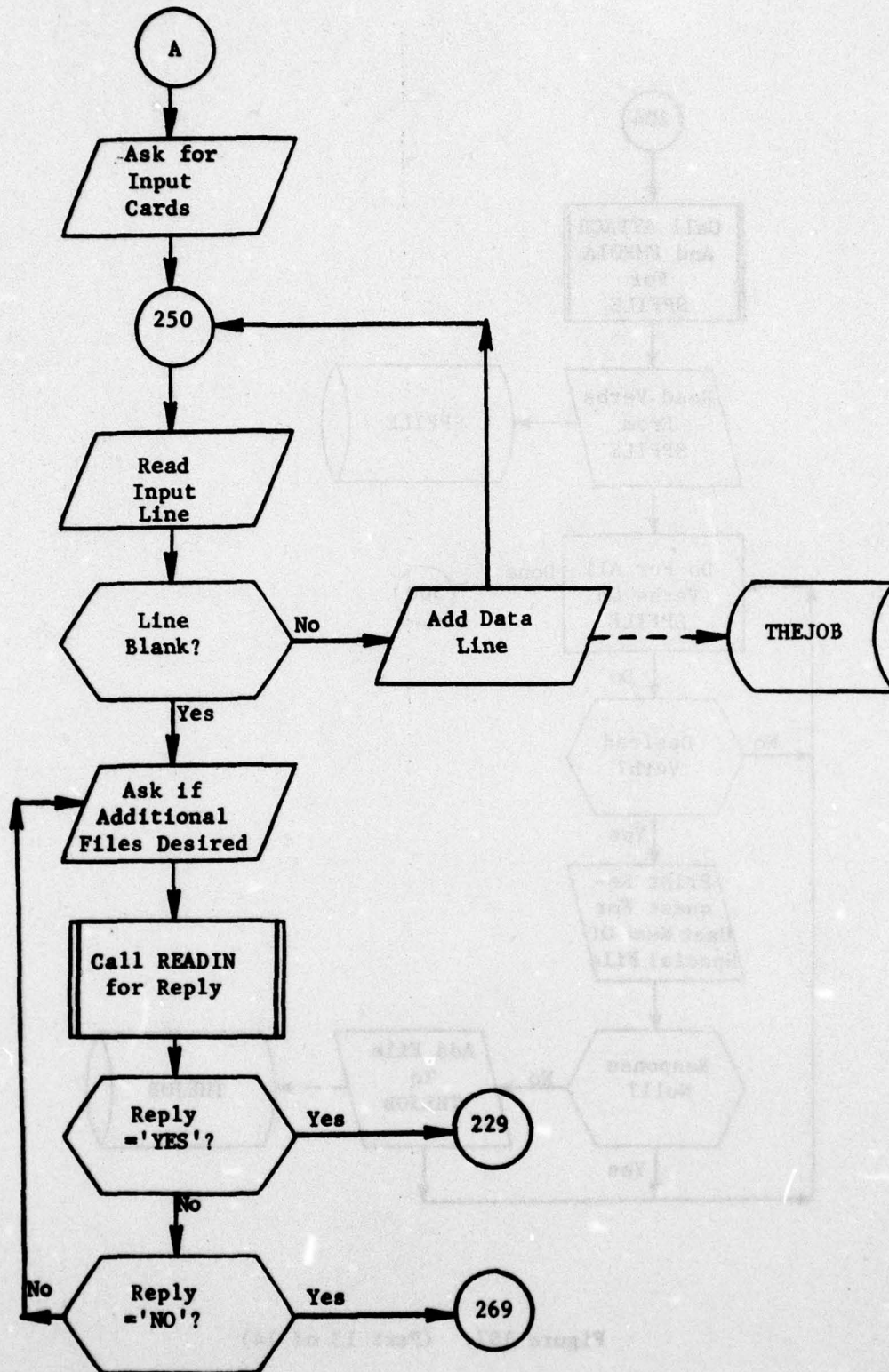


Figure 197. (Part 12 of 14)

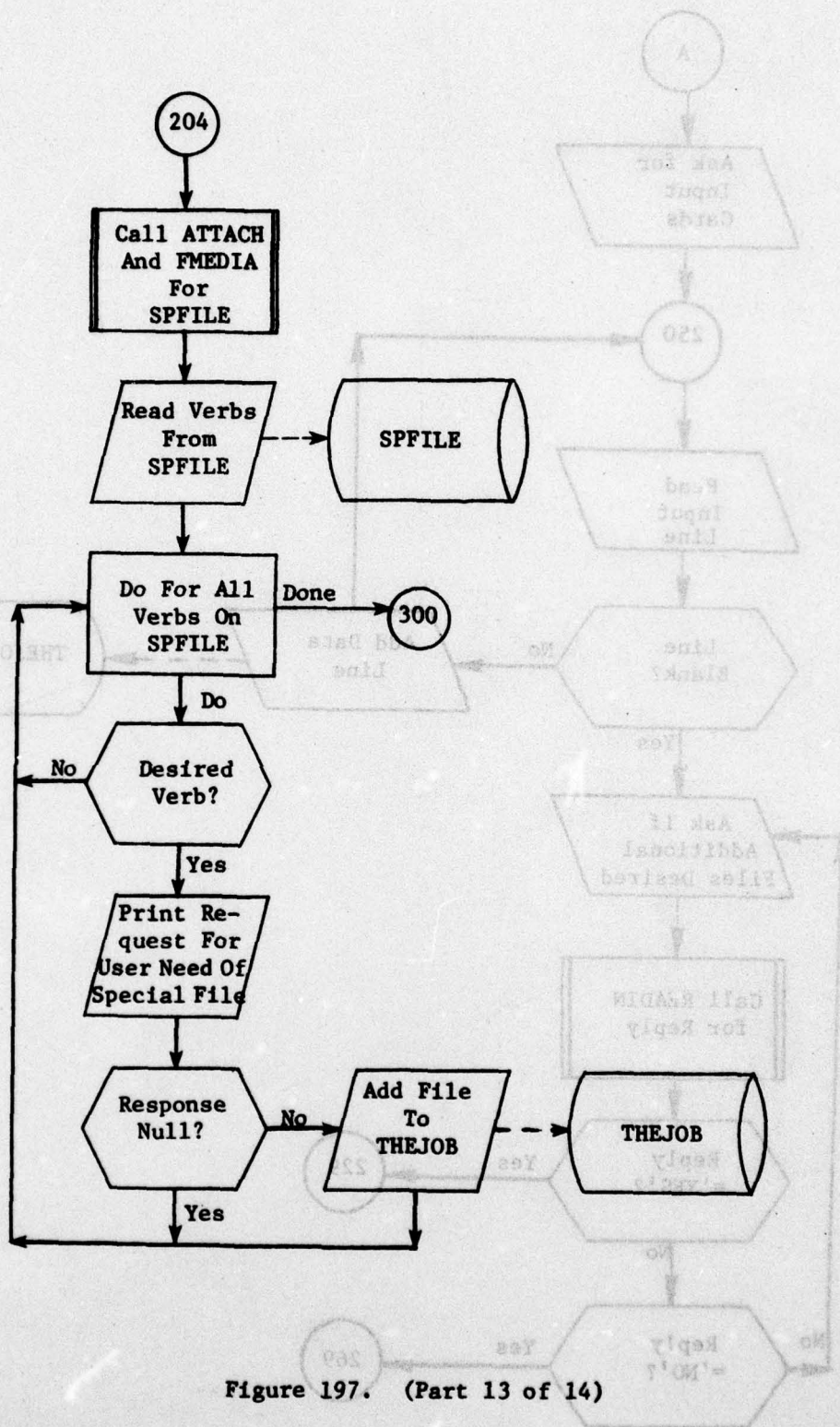


Figure 197. (Part 13 of 14)

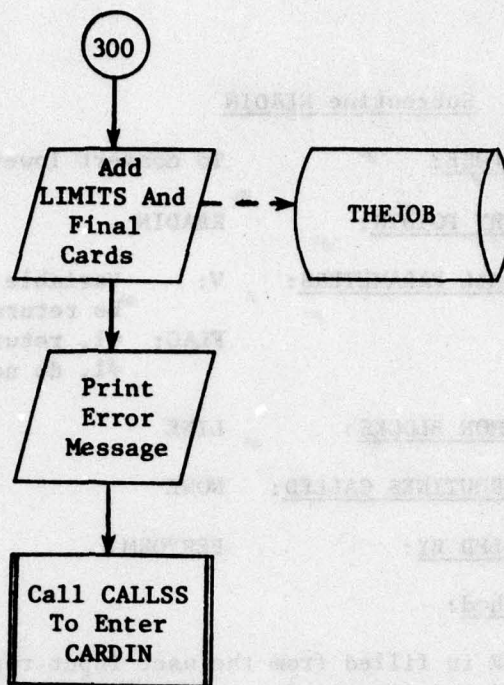


Figure 197. (Part 14 of 14)

C.8 Subroutine READIN

PURPOSE: To convert lower case to upper case

ENTRY POINTS: READIN

FORMAL PARAMETERS: V: Variable in which first eight characters may
be returned
FLAG: =1, return first eight characters in V.
#1, do not alter V.

COMMON BLOCKS: LINE

SUBROUTINES CALLED: NONE

CALLED BY: PERFORM

Method:

LINE is filled from the user input replay. Each character is then examined and converted to upper case if it is lower case. FLAG is checked and if equal to 1, the first eight positions of LINE are encoded into V.

Subroutine READIN is illustrated in figure 198.

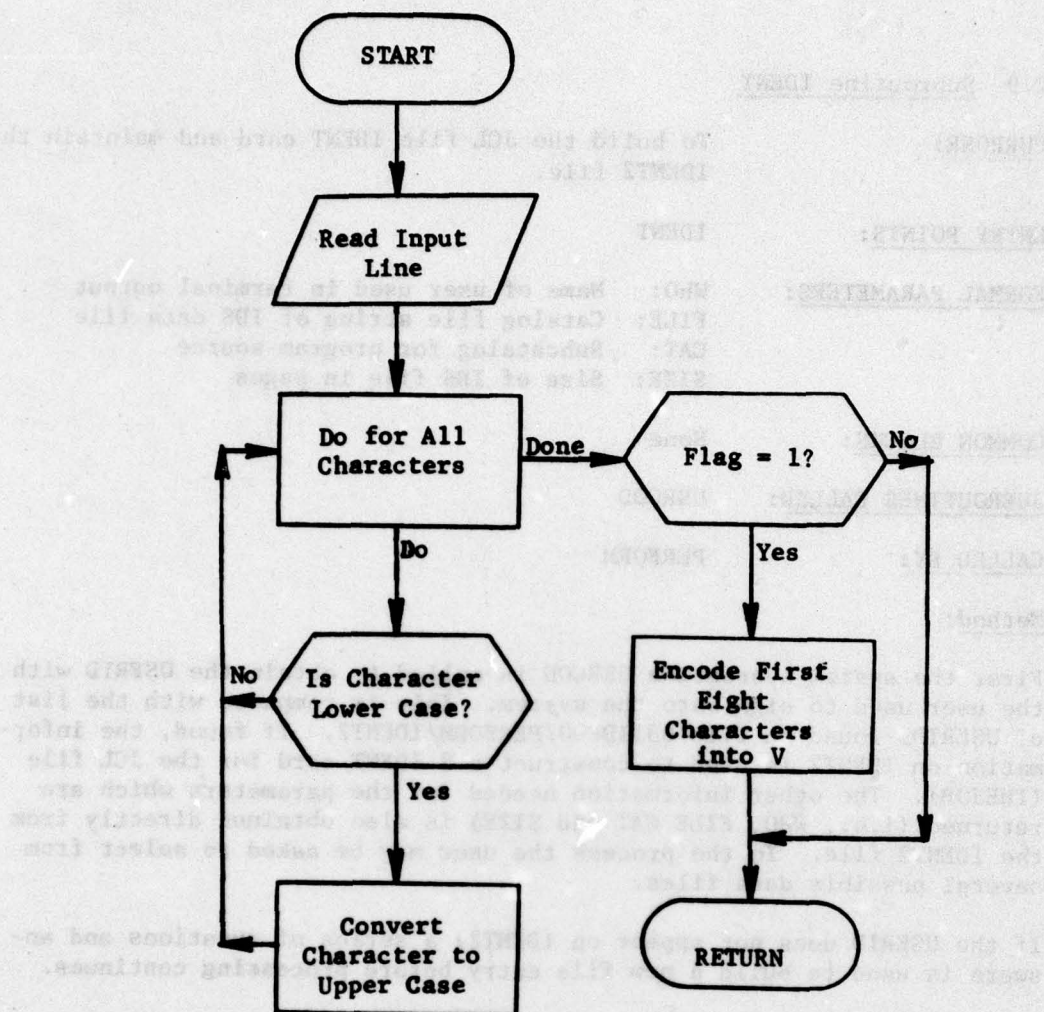


Figure 198. Subroutine READIN

C.9 Subroutine IDENT

PURPOSE: To build the JCL file IDENT card and maintain the IDENT2 file.

ENTRY POINTS: IDENT

FORMAL PARAMETERS: WHO: Name of user used in terminal output
FILE: Catalog file string of IDS data file
CAT: Subcatalog for program source
SIZE: Size of IDS file in pages

COMMON BLOCKS: None

SUBROUTINES CALLED: USRCOD

CALLED BY: PERFORM

Method:

First the system subroutine USRCOD is called to obtain the USERID with the user used to sign onto the system. This is compared with the list of USERIDs found on file 631IDP00/PERFORM/IDENT2. If found, the information on IDENT2 is used to construct a \$ IDENT card for the JCL file (THEJOB). The other information needed for the parameters which are returned (i.e., WHO, FILE CAT and SIZE) is also obtained directly from the IDENT2 file. In the process the user may be asked to select from several possible data files.

If the USERID does not appear on IDENT2, a series of questions and answers is used to build a new file entry before processing continues.

Subroutine IDENT is illustrated in figure 199.

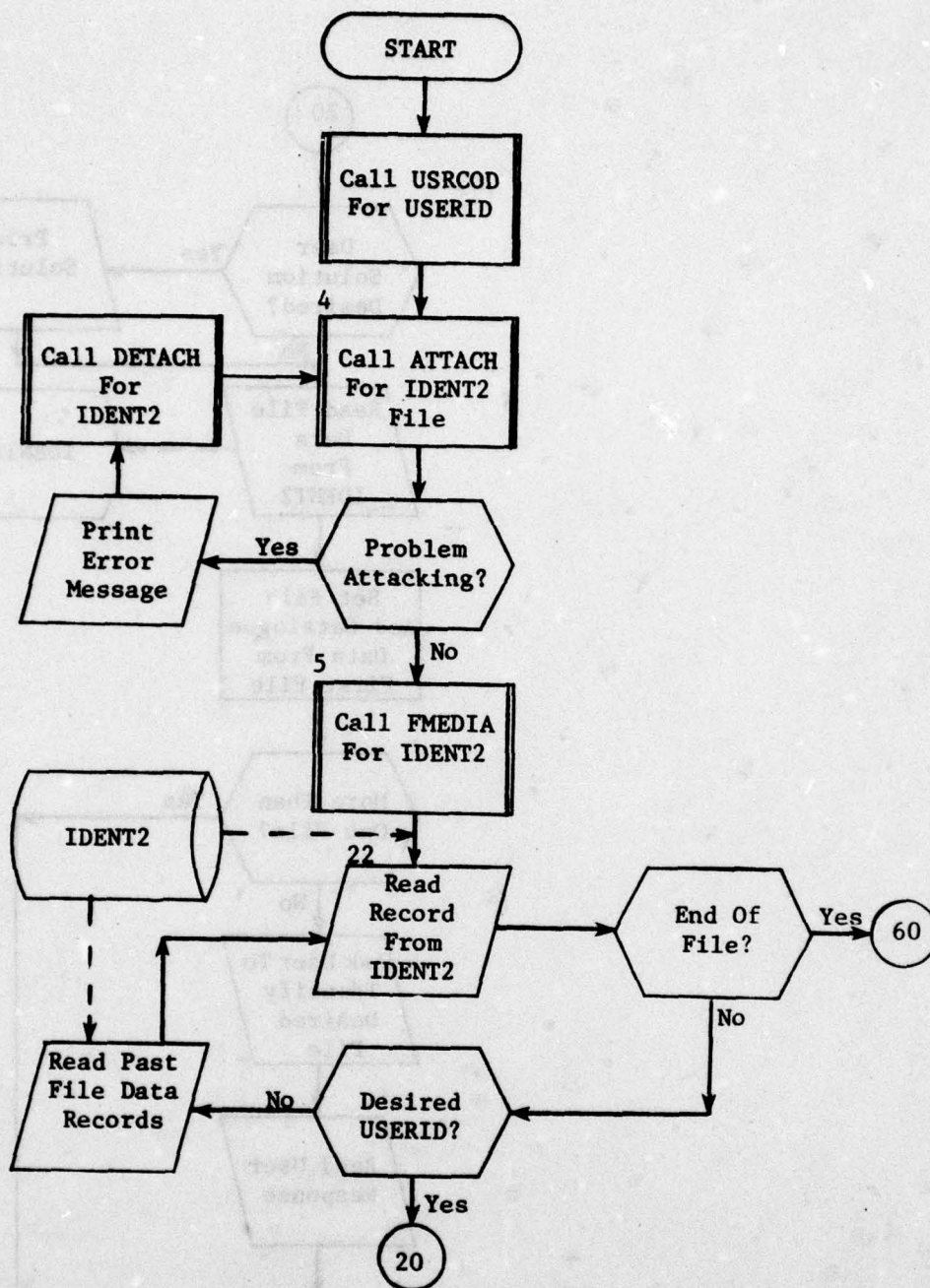


Figure 199. Subroutine IDENT (Part 1 of 4)

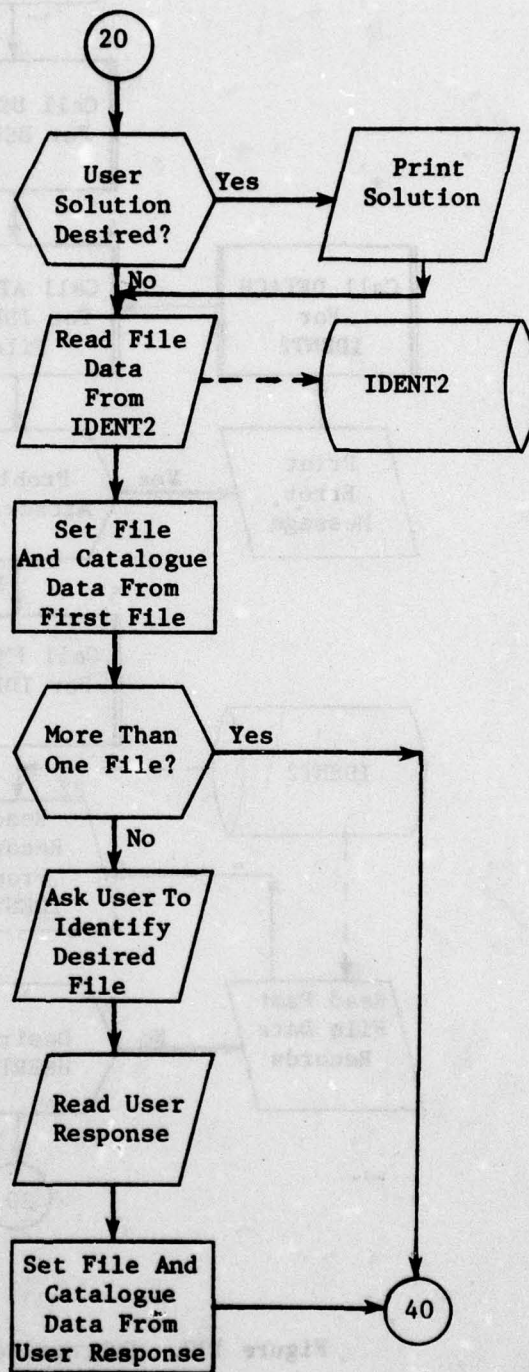


Figure 199. (Part 2 of 4)

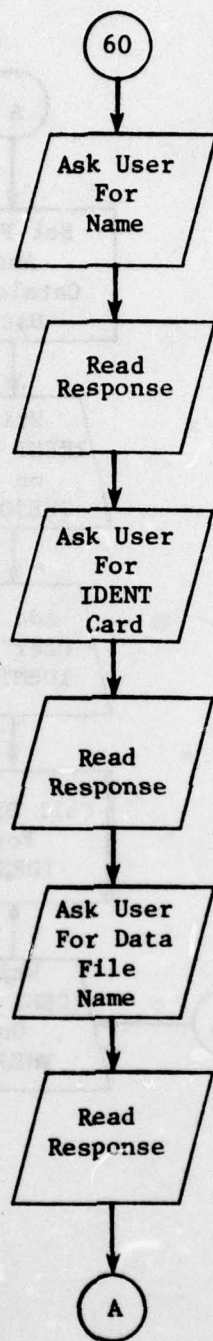


Figure 199. (Part 3 of 4)

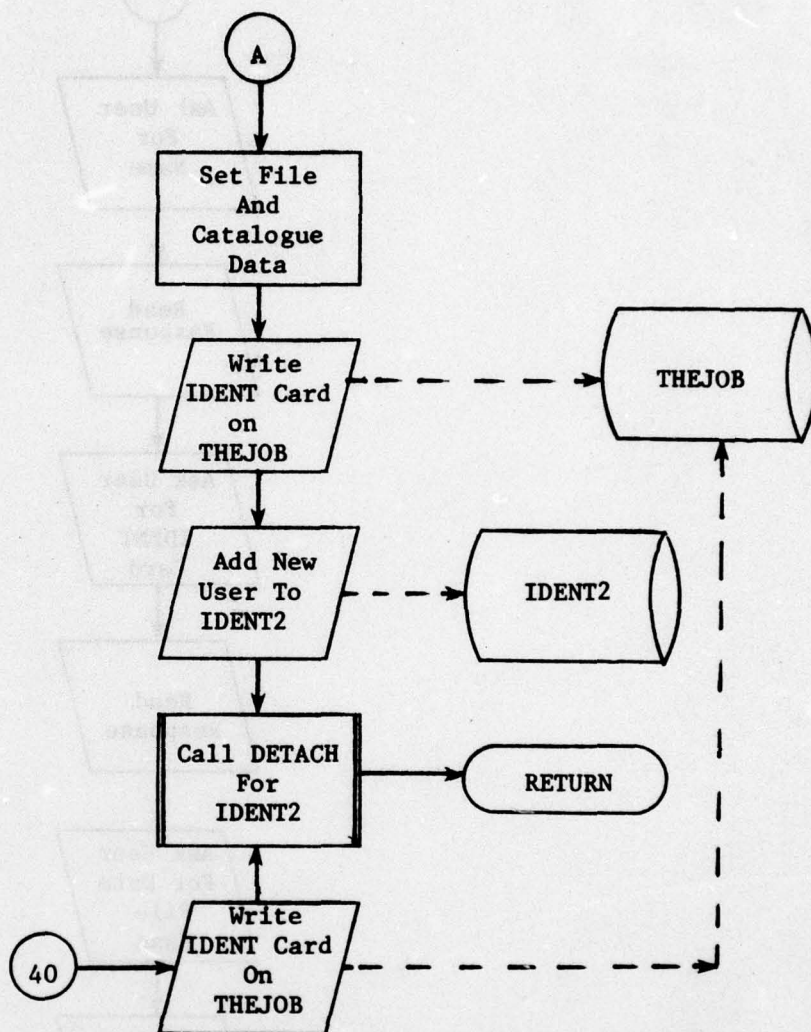


Figure 199. (Part 4 of 4)

DISTRIBUTION

<u>Addressee</u>	<u>Copies</u>
CCTC Codes	
Technical Library (C124)	3
C124 (Stock)	6
C313	1
C314	17
C600	1
DCA Code	
205	1
EXTERNAL	
Chief, Studies, Analysis and Gaming Agency, OJCS	
ATTN: SFD, Room 1D957, Pentagon, Washington, DC	
20301	2
Chief of Naval Operations, ATTN: OP-96C4, Room 4A478,	
Pentagon, Washington, DC 20350	2
Commander-in-Chief, North American Air Defense Command	
ATTN: NPXYA, Ent Air Force Base, CO 80912	2
U. S. Air Force Weapons Laboratory (AFSC)	
ATTN: AFWL/SUL (Technical Library),	
Kirtland Air Force Base, NM 87117	1
Director, Strategic Target Planning, ATTN: (JPS), Offutt	
Air Force Base, NE 68113	2
Defense Documentation Center, Cameron Station,	
Alexandria, VA 22314	12
	<u>50</u>